عنوان پروژه:

بررسی زبان های برنامه نویسی و مقایسه آنها با یکدیگر و

ارائه راهکار برای انتخاب زبان برنامه نویسی مناسب

عنوان درس:فن آوری اطلاعات

نام استاد:دکتر مسعود حجاریان

تهیه کننده: آزاده متحدین

تاریخچه زبان های برنامه نویسی:

زبان های برنامه نویسی دارای نسل های مختلفی می باشند که به ترتیب عبارتند از:

نسل اول: زبان ماشین که در اواخر دهه ۱۹۴۰ بوجود آمد : در این زبان که تنها زبان قابل فهم برای کامپیوتر می باشد از ارقام صفر و یک به عنوان علایم اولیه استفاده می شود و ارقام صفر و یک در حقیقت الفبای این زبان محسوب می شوند و جهت ایجاد کدهایی برای دستورالعمل ها بکار می روند،به طوری که هر دستورالعمل به صورت رشته ای از صفر و یک ها نوشته می شود.

نسل دوم: زبان اسمبلی که در اوایل دهه ۱۹۵۰ بوجود آمد : این زبان در واقع همان زبان ماشین است، با این تفاوت که جهت ساده نمودن کار برنامه نویس، کدهای سمبلیکی به نام کد نیمانیک در اوایل سال های ۱۹۵۰ بوجود آمد که در آن، از حروف برای کد گذاری کدهای زبان ماشین استفاده شد که این کدها در ریزپردازنده های مختلف با یکدیگر تفاوت دارند. این کدها توسط نرم افزار های به خصوصی بنام اسمبلر به زبان ماشین تبدیل می گردند تا قابل درک برای ماشین باشند.

نسل سوم: زبان های سطح بالا که در اواخر دهه ۱۹۵۰ بوجود آمدندو اولین زبان این نسل فرترن بود. به این زبان ها زبان های رویه ای یا رویه گرا نیز گفته می شود زیرا برنامه نویس باید چگونگی این عملیات را تشریح نماید.

نسل چهارم: (اواسط دهه ۷۰) این زبان ها بسیار شبیه به زبان های طبیعی می باشند و برنامه نویس بوسیله برنامه هایی که به این زبان ها می نویسد به روشی ساده تر از زبان های دیگر با کامپیوتر ارتباط برقرار می نماید، گویی در حال صحبت کردن معمولی با کامپیوتر می باشد. به این زبان ها زبان های غیررویه ای نیز گفته می شود زیرا برنامه نویس بدون تشریح چگونگی عملیات، خواسته خود را مطرح می کند. یعنی به کامپیوتر می گوید چه می خواهد، ولی چگونگی انجام عملیات را نمی گوید. به عنوان مثال از این نسل می توان به زبان ADAاشاره نمود. به این زبان ها، زبان های فوق بالا نیز گفته می شود.

در کل زبان های برنامه سازی به دو دسته تقسیم می شوند:

۱ -زبان های سطح پایین Low level languages

۲ -زبان های سطح بالا High level languages

زبان های سطح پایین

زبان هایی هستند در سطح ماشین و به دور از زبان طبیعی و محاوره ای انسان. این زبان ها وابسته به ماشین و سخت افزار هستند، بطوری که هر میکروپروسسور زبان خاص خود را داراست. کار کردن با این زبان ها مشکل است و خطا یابی و بررسی برنامه ها به سهولت امکان پذیر نیست، ولی به علت نزدیکی به ماشین، برنامه های نوشته شده به این زبان ها با سرعت بالایی اجرا می شوند. زبان های سطح پایین به دو دسته تقسیم می شوند:

۱–زبان ماشین .

۲–اسمبلی .

زبان های سطح بالا

زبان هایی هستند نزدیک به زبان طبیعی و محاوره ای انسان که در آن ها از علایم، حروف و کلمات آشنا و بکار رفته در زبان طبیعی استفاده می شود. این زبانها احتیاج به ترجمه و تصویر دارند تا قابل درک برای کامپیوترها شوند که این امر بوسیله نرم افزار های بخصوصی بنام کامپایلر و مفسر انجام می گیرد. زبان های سطح بالا وابسته به ماشین ، سخت افزار نیستند و با اندکی تغییر در کلیه کامپیوترها قابل اجرا می باشند. البته شرط قابل درک بودن این زبان ها برای کامپیوترها وجود مفسر با کامپایلر می باشد. کارکردن با این زبان ها آسان تر است و نسبت به زبان های سطح پایین خطایابی و بررسی برنامه ها راحت تر انجام می گیرد.

تولید زبان های سطح بالا از اواسط دهه ۱۹۵۰ آغاز گردید و متداولترین آن ها عبارتند از:

FORTRAN,COBOL,PL/1,BASIC,LOGO,PSCAL,C

C

زبان C در آزمایشگاه BELL در اوایل دهه ۱۹۷۰ به منظور تکمیل و باز نویسی نسخه اولیه سیستم عامل UNIX طراحی شد و امروزه نسخ مختلفی از زبان C بوجود آمده است. گر چه C یک زبان سطح بالا است ولی غالبا به عنوان زبان برنامه نویسی سیستم و یا برای رفع نیازهایی که در گذشته به کمک زبان اسمبلی برطرف می شدند استفاده می شود. همچنین بسیاری از نرم افزارهای اساسی کامپیوتر به این زبان نوشته می شوند. فراگیری این زبان برای مبتدیان کار دشواری است.

LOGO

این زبان توسط سیمور پاپرت در دهه ۱۹۶۰ در دانشگاه MIT عرضه شد. گرچه این زبان جهت استفاده دانشجویان به منظور کارهای علمی طراحی گردید، لیکن آن را به عنوان اولین زبان آموزشی جهت پرورش مهارت و خلاقیت بچه ها می شناسند.

رسم خطوط گرافیکی، کار روی رنگ ها، ایجاد تصاویر متحرک در این زبان بسادگی انجام می شود.

LISP

توسط جان مک آرتی در سال ۱۹۶۰–۱۹۵۹ به منظور پشتیبانی تحقیق در زمینه هوش مصنوعی (AI) ارائه گردید و روی داده های غیر عددی کار می نماید و جهت برنامه نویسی در محیط AUTO CAD نیز مورد استفاده قرار می گیرد.

ALGOL

این زبان در سال ۱۹۵۸ معرفی گردید و یک زبان علمی می باشد. نسخ مختلفی از ALGOL تا کنون عرضه شده است که از جمله می توان ALGOL 68 را نام برد. در آمریکا معمولا از FORTRAN بجای ALGOL استفاده می شود، ولی در اروپا این زبان از محبوبیت ویژه ای برخوردار است.

PASCAL

این زبان که به افتخار بلز پاسکال دانشمند فرانسوی قرن هفدهم میلادی، پاسکال نامگذاری شده است در اواخر سال های ۱۹۶۰ و اوایل 1970توسط پروفسور نیکلاس ویرث در انستیتو فنی فدرال سوئیس مطرح گردید .این زبان از قدرت بالایی در اجرای امور علمی و تجاری برخوردار است و در بسیاری از مدارس و کالج های دنیا جهت آموزش برنامه نویسی تدریس می گردد و در سال ۱۹۸۳ توسط سازمان استاندارد ملی آمریکا بصورت استاندارد در آمد.

ADA

این زبان به افتخار نام دختر لرد بایرون که همکار چالز بابیج در زمینه طرح ماشین تحلیلی بود، ADAنامیده شد. خانم ADA را بخاطر برنامه هایش به عنوان اولین برنامه نویس در جهان می شناسند.

این زبان به منظور سرویس های نظامی در وزارت دفاع آمریکا تهیه گردید. در سال ۱۹۷۵ وزارت دفاع آمریکا تحقیقاتی را پیرامون طراحی یک زبان عمومی که مورد استفاده فروشندگان کامپیوتر و برنامه نویسان نظامی باشد آغاز کرد که ماحصل کار آن ها زبان ADA بود که در سال ۱۹۸۰ عرضه گردید. این زبان در سال 1983توسط سازمان ملی استاندارد آمریکا به صورت استاندارد در آمد.

Basic

بیسیک به معنی زبان همه منظوره برای افراد مبتدی می باشد. این زبان به خاطر ساختار ساده ای که دارد از محبوبیت فوق العاده ای در جهان برخوردار است. در سیستم های محاوره ای و اشتراک زمانی استفاده می شود. یک زبان محاوره ای امکان ارتباط مستقیم بین انسان و کامپیوتر را در حین اجرای برنامه فراهم می نماید. یک فرد مبتدی که آشنایی چندانی با کامپیوتر ندارد پس از مدت کوتاهی می تواند دستورهای این زبان را فرا گرفته و اقدام به نوشتن برنامه بنماید. وارد کردن داده ای ورودی بسیار ساده بوده و برنامه نویس لازم نیست نگران دستورهای (فرمت) خروجی برنامه باشد زیرا فرمت های خروجی قابل استفاده توسط این زبان در اختیار است. همچنین ایجاد تغییرات و اضافه کردن داخل برنامه بیسیک بسادگی انجام می شود. به خاطر سادگی این زبان، BASICدر اولین میکرو کامپیوترها مورد استفاده قرار گرفت و تا کنون نیز محبوب ترین زبان سطح بالای مورد استفاده در این سیستم های شخصی برای آموزش نو آموزان می باشد. زبان بیسیک بین سال های ۱۹۶۳ و ۱۹۶۴ توسط پروفسور جان کمنی و توماس کورتز در کالج دارتموث بوجود آمد و هدف آن ها از

ایجاد زبان بیسیک این بود که کلیه دانشجویان رشته های مختلف بتوانند آن را بسادگی فرا گیرند. علیرغم اینکه در بیسیک اولیه از دستورات معین و محدودی استفاده می شد، لیکن سازندگاه کامپیوتر دستورات متعددی را به آن افزودند و از نظر سخت افزاری امکانات کامپیوتر خود را افزایش دادند تا بتوانند با سایر سازندگان کامپیوتر رقابت نمایند، لذا امروزه نسخه های متعددی از بیسیک وجود دارد و سازمان استاندارد آمریکا نسخه ای از آن را بنام نسخه پایه در سال ۱۹۷۸ ارائه نمود. استاندارد فوق به حدی ساده است که نسخه های گسترش یافته آن از قبیل GWBASIC, QBASIC, TURBO BASIC,VISUAL BASIC قابل دسترسی است. اینک از دانش آموزان مدارس تا مهندسان هواپیما از این زبان استفاده می نمایند. همچنین این زبان در امور تجاری و مدیریت کاربرد دارد.

COBLOL

کوبول به معنی زبان تجاری می باشد که برای پردازش فایل ها بوجود آمد و هم اکنون برای کارهای تجاری با حجم زیاد مورد استفاده قرار می گیرد. در سال ۱۹۵۹بسیاری از نمایندگان دولت آمریکا و سازندگان و استفاده کنندگان کامپیوتر و دانشگاه ها گرد هم آمدند تا زبان مناسب برای پردازش فایل ها را بوجود آورند. حاصل کار آن ها در ژانویه سال ۱۹۶۰ به اتمام رسید. مشخصات این زبان چند ماه بعد توسط سازمان انتشارات دولتی به ثبت رسید و در سال ۱۹۶۱ کامپایلر زبان کوبول برای امور تجاری عرضه شد. سازمان ANSI استانداردی برای زبان کوبول در سال ۱۹۶۸ تهیه کرد و در سال ۱۹۷۴ نسخه جدیدی از آن نیز عرضه شد. زبان کوبول بهتر از سایر زبان های برنامه نویسی قادر به انجام عملیات بر روی کاراکتر های الفبایی از قبیل نام، آدرس و سایر مشخصات دیگر می باشد و محدودیت آن این است که برای انجام عملیات پیچیده ریاضی مناسب نمی باشد.

PL/1

همانطور که ملاحظه نمودید زبان های اولیه از قبیل فرترن و کوبول به منظور حل مسائل علمی، تجاری تهیه گردیدند. اما در اوایل دهه ۱۹۶۰ شرکت IBM و یک کمیته از استفاده کنندگان IBM 360 کار خود را بر روی زبانی که قابلیت فرترن و کوبول را تواما داشته باشد آغاز نمودند که PL/1 نامیده شد و در اواسط دهه ۱۹۶۰ کار تهیه این زبان به پایان رسید.

PL/1نیز مانند یک زبان علمی از برخی تکنیک های فرترن و کوبول بهره جست و سازمان استاندارد آمریکا (ANSI) در سال ۱۹۷۶ استانداردی برای آن تهیه کرد. علیرغم اینکه PL/1زبان پرقدرتی می باشد و به این منظور ساخته شد که جایگزین فرترن و کوبول گردد لیکن از آنجایی که فراهم نمودن تکنیک هایی که قادر به انجام امور علمی و تجاری باشد مشکل است، لذا PL/1 موفقیت مورد نظر را کسب ننمود. با توجه به اینکه فراگیری این زبان نیز ساده نیست می رود که به تدریج به دست فراموشی سپرده شود.

(Formula Translation) FORTRAN

فرترن اولین زبان سطح بالا است که تولید آن در سال ۱۹۵۴ به سرپرستی جان باکوز به منظور ایجاد یک زبان علمی در شرکت IBM شروع و در سال ۱۹۵۷ روی IBM 704 معرفی گردید که بالغ بر ۲/۵ میلیون دلار هزینه برداشت. با استفاده از این زبان حل معادلات ریاضی بسیار آسان گردید و بسیار مورد استقبال قرار گرفت. این زبان در اکثر کامپیوترهای بزرگ و کوچک مورد استفاده قرار می گیرد و همین استقبال فوق العاده سبب شد تا کار تهیه استاندارد در سال ۱۹۶۲ برای آن شروع شود که یکی از آن ها را نسخه پایه و دیگری را نسخه کامل یا گسترش یافته می نامند. استاندارد زبان فرترن در سال ۱۹۶۶ مورد پذیرش سازمان استاندارد آمریکا قرار گرفت و این اولین زبانی بود که به صورت استاندارد درآمد. برنامه هایی که به این زبان در یک کامپیوتر نوشته می شود معمولا به سادگی در سایر کامپیوترها نیز قابل استفاده می باشد. فرترن نیز از دستورات ورودی، خروجی، محاسباتی، منطقی / مقایسه ای و سایر دستورات اساسی از قبیل READ، WRITE، GOTOو STOP همانطور که از این دستور ها در زبان انگلیسی انتظار می رود استفاده می شود.

زبان فرترن قابلیت حل مسایل ریاضی و آماری را دار می باشد، لذا بسیاری از برنامه های این مقوله به این زبان نوشته می شوند. از ضعف های این زبان این است که دنبال کردن منطق برنامه مشکل تر از سایر زبان های سطح بالا می باشد و این زبان برای پردازش فایل ها نمی باشد، لذا برای پردازش فایل ها و استفاده در امور تجاری زبان سطح بالای دیگری بوجود آمد به نام کوبول.

دسته بندی زبان های برنامه نویسی:

Functional Languages

زبان های ساخت یافته به زبان هایی می گویند که از ساختار منظم که توسط توابع ایجاد شده اند برخوردار هستند . مثلا برای نوشتن برنامه ای که می خواهد دو عدد را باهم جمع کند تابعی به نام add بنویسیم که دو عدد را بگیرد و با هم جمع کند و این تابع را در برنامه های دیگرمان هم استفاده کنیم. برخی از زبان های ساخت یافته در زیر نوشته شده اند:

Haskell: یک زبان برنامه نویسی کاملا ساخت یافته.

ML: یک زبان برنامه نویسی استاندارد که نگارش های مختلفی دارد.

Erlang: یک زبان ساخت یافته با کاربرد صنعتی.

J&K: دو زبان برنامه نویسی با قابلیت های آرایه ای قوی.

APL: یک زبان برنامه نویسی ساخت یافته بر پایه آرایه ها.

LISP: یک زبان برنامه نویسی ساخت یافته. البته از این زبان به بعد برنامه نویس شی گرا مطرح شد.

C: این زبان به عنوان اولین زبان برنامه نویسی سیستمی مطرح است.

زبان هایی مثل Pascal و Basic هم از این گروه هستند.

در ۲۰ سال اخیر کمیته برنامه نویسی ساخت یافته بجای ارائه زبان جدید به ارتقای زبان هایی مثل ML و LISP پرداختند.

Object Oriented Language

زبان های شی گرا زبان هایی هستند که بر پایه اشیاء ساخته ونوشته می شوند همه چیز در این زبان ها شی محسوب می شود. و هم عرض با این زبان ها نوعی برنامه نویسی به نام OOP یا Object Oriented Programming بوجود آمده این زبان ها در واقع نسل جدید زبان های برنامه نویسی هستند:

Simula: اولین زبان برنامه نویسی oo که در سال ۱۹۶۰ عرضه شد.

++C: این زبان در حقیقت ترکیبی از برنامه نویسی سیستمی و شی گرا است.

Perl: یک زبان برنامه نویسی تحت UNIX که برای ساخت وب سایت های پویا بکار می رود.

PHP: زبان برنامه نویسی سمت سرور که در سالهای اخیر محبوبیت زیادی کسب کرده است.

Java: زبان برنامه نویسی محصول شرکت Sun

تعداد زبان های برنامه نویسی واقعا زیاد است ASP و VB و ... نیز از این جمله اند.

Scripting language

زبان های متنی زبان هایی هستند که نحوه نگارش آنها به زبان طبیعی نزدیک است.

این گونه زبان ها معمولا پیچیدگی های انواع دیگر را ندارند و می توان گفت ساده تر هستند.

TCL: یا Tickle پدر بزرگ زبان های متنی

Perl: یک زبان متنی است که قابلیت شی گرا دارد

Java Script: محبوبترین زبان برنامه نویسی متنی در حال حاضر برای برنامه نویسی سمت مشتری در صفحات وب .

Python: زبان برنامه نویسی قدرتمند با قابلیت شی گرایی بسیار قوی برای برنامه نویسی سمت مشتری در صفحات وب .

Logical Language

زبان های برنامه نویسی منطقی به نوعی سردم دار سبک جدیدی از برنامه نویسی هستند که در علوم مختلف کاربرد دارد

Prolog: اولین زبان برنامه نویسی متنی که در سال۱۹۷۲ ایجاد شد.

Mercury: زبان برنامه نویسی منطقی دیگر ...

ویژگی های زبان برنامه نویسی دلفی و موارد کاربرد:

Delphi یکی از موفق ترین ابزارهای توسعه نرم افزار برای ویندوز در دهه گذشته بوده است. پس از شکست ویژوال بیسیک ۶ در مقابل دلفی ۷ در سال ۲۰۰۲ و فراگیر شدن محیط توسعه شرکت بورلند در میان توسعه دهندگان ویندوز، مایکروسافت به فکر ساخت یک محیط توسعه مجتمع مناسب افتاد. در آن زمان دلفی دوران طلایی خود را می گذراند. دلفی ۷ در اکثر شرکت های نرم افزاری دنیا و حتی در همین ایران خودمان به اولین انتخاب برای توسعه نرم افزارهای تحت ویندوز تبدیل شده بود. این محبوبیت با آمدن ویژوال استادیو ۲۰۰۳ هنوز هم ادامه داشت. مایکروسافت با ارائه دات نت فریم ورک ۲.۰ و ویژوال استادیو ۲۰۰۵ که به برنامه نویسان امکان توسعه نرم افزارهای تحت ویندوز و تحت وب را به بهترین شکل می داد توانست در این رقابت دلفی را پشت سر بگذارد.

ارائه دلفی ۲۰۰۵ و ۲۰۰۶ به ضعیف ترین شکل ممکن توسط شرکت بورلند، اکثر توسعه دهندگان دلفی را از آن نا امید کرد و به فکر مهاجرت به یک محیط توسعه دیگر مثل دات نت و جاوا ترغیب نمود. پس از آن شرکت بورلند توسعه دلفی را به یکی از شرکت های زیرمجموعه خود به نام CodeGear واگذار کرد. با ارائه نسخه ۲۰۰۷ توسط این شرکت روح تازه ای به پیکر نیمه جان دلفی دمیده شد و بسیاری از برنامه نویسان ویندوز را به استفاده از آن تشویق نمود. در سال ۲۰۰۸ بورلند شرکت CodeGear را به شرکت Embarcadero technologies فروخت تا نسخه ۲۰۰۹ دلفی با نام Embarcadero Delphi 2009 به بازار وارد شود. نسخه ۲۰۰۹ موفق ترین انتشار دلفی پس از دلفی ۷ بود.

با خواندن این مقدمه نسبتاً طولانی و خلاصه ای از تاریخچه دلفی و با ارائه دلفی ۲۰۱۰ توسط شرکت Embarcadero این سئوال پش می آید که **آیا هنوز هم دلفی محیط توسعه قابل اعتماد و مناسبی برای انجام پروژه های نرم افزاری هست یا خیر؟**

ساده ترین جواب به این سئوال جواب "بله" است. نسخه ۲۰۱۰ دلفی انتخاب مناسبی برای توسعه نرم افزارهای تحت ویندوز است. این محیط زمان توسعه پروژه شما را کاهش می دهد و با ابزارها و کامپوننت های آماده ای که دارد تولید یک برنامه تحت ویندوز را برای هر برنامه نویسی آسان می کند. نسخه ۲۰۱۰ دلفی دارای خصوصیات جدید و ویژگی های مناسبی است. برنامه های دلفی بر خلاف برنامه های مبتنی بر دات نت و جاوا به هیچ پیش نیازی وابسته نیستند و خروجی دلفی یک فایل اجرایی stand-alone است که به مراتب سریع تر از خروجی های محیط های ویژوال استادیو و جاوا اجرا می شود.

دلفی ۲۰۱۰ از تمامی قابلیت های جدید ویندوز ۷ مثل direct 2D، صفحه لمسی و اشاره ای پشتیبانی می کند. اکثر کامپوننت های VCL در این نسخه طوری تغییر کرده اند که با ویندوز ۷ سازگاری کامل داشته باشند. Delphi Prism هم قابلیت نوشتن برنامه های مبتنی بر دات نت را به زبان دلفی به برنامه نویسان می دهد.

اما اینکه واقعاً دلفی ۲۰۱۰ می تواند انتخاب مناسبی برای شما باشید یا خیر، کاملاً بستگی به نوع پروژه و موقعیت شما دارد. به طور مثال اگر می خواهید یک نرم افزار مبتنی بر پایگاه داده ها تحت ویندوز و به صورت کلاینت/سرور بنویسید، شک نکنید که دلفی ۲۰۱۰ بهترین گزینه برای شماست. اما اگر می خواهید یک برنامه تحت وب که از وب سرویس هم استفاده می کند و ساختار پیچیده ای دارد تولید کنید، دلفی ۲۰۱۰ نمی تواند انتخاب خوبی باشد. در کل برای توسعه نرم افزارهای مبتنی بر Win32 دلفی می تواند بهترین انتخاب باشد. اما برای توسعه مبتنی بر دات نت یا وب نمی تواند انتخاب مناسبی باشد.

نکته دیگری که وجود دارد این است که در ایران موقعیت های شغلی برای دلفی به ندرت پیدا می شود. این در حالی است که هزینه نرم افزارهایی که با دلفی نوشته می شوند به مراتب پایین تر از نرم افزارهایی است که با ابزارهای مایکروسافت تولید می شوند. اما به علت نبود قانون کپی رایت در ایران، اکثر شرکت های نرم افزاری سعی می کنند از گران ترین تکنولوژی ها و ابزارها برای تولید نرم افزار خود استفاده کنند. هر چند در خارج از کشور هم وضع به همین صورت است و اکثر شرکت ها از ابزارهای مایکروسافت و اوراکل استفاده می کنند اما به علت وجود قانون کپی رایت شرکت ها و افرادی زیادی نیز پیدا می شوند که به علت هزینه پایین تر دلفی از آن برای توسعه پروژه های خود بهره می برند.

در آخر به این نکته توجه داشته باشید که هر ابزاری را برای هدفی ساخته اند. اینکه یک برنامه نویس فقط به یک ابزار و یک زبان برنامه نویسی وابسته باشد اصلاً خوب نیست. هنوز یک ابزار و یک زبان همه منظوره که بتواند پاسخگوی تمام نیازهای برنامه نویسی باشد وجود ندارد. شما باید بر اساس موقعیت و نیاز خود از این ابزارها و زبان ها استفاده کنید. دلفی ابزاری عالی و مناسب برای اضافه کردن به ابزارهای برنامه نویسی شما خواهد بود!

بورلند دلفی ۷ محیط برنامه نویسی بصری (ویژوال) و کاملا شیء گرا برای توسعه برنامه های ۳۲بیتی مبتنی بر سیستم عامل های ویندوز و لینوکس می باشد. با استفاده از دلفی می توانید برنامه های کاربردی بسیار قوی و انعطاف پذیر ایجاد کرده و آن ها را در ویندوز و لینوکس انتشار دهید.

محیط دلفی ابزارهای سریع و ساده بسیاری در اختیار برنامه نویسان قرار میدهد. محیط دلفی در اصطلاح **RAD** خوانده می شود. این کلمه مخفف **Rapid Application Development** و به معنی توسعه کاربردی سریع است. دلفی شامل ابزارهای سریع طراحی برنامه، ویزاردهای برنامه نویسی، قالب های از پیش ساخته شده و کتابخانه ای از اجزای نرم افزاری است.
دلفی شامل دو کتابخانه مجزا برای ویندوز و لینوکس است :

- **The Visual Component Library – VCL** : این کتابخانه شامل اجزایی است که API های ویندوز را در خود کپسوله کرده اند.

- **The Borland Component Library for Cross-platform – CLX** :

که این کتابخانه اجزای Qt Library را خود کپسوله نموده اند. (این نوع از برنامه ها در ویندوز و لینوکس قابل اجرا هستند)

در این بخش به شرح مختصری از قابلیت های محیط دلفی ۷ می پردازیم.

محیط توسعه مجتمع (IDE) :

وقتی دلفی را باز می کنید بلافاصله محیط توسعه مجتمع یا به اختصار IDE دلفی در مقابل شما نمایان می شود. این IDE شامل تمامی ابزارهای مورد نیاز برای طراحی، توسعه، تست، و انتشار برنامه می باشد.

- **Form Designer**: برای طراحی رابط کاربری برنامه
- **Component palette** : برای دسترسی به کامپوننت های بصری و غیر بصری قابل استفاده در برنامه
- **Object Inspector** : برای مشاهده و مدیریت خصوصیات اشیاء و رویدادهای مربوط به هر یک
- **Object TreeView** : برای نمایش و مدیریت روابط منطقی میان اجزای برنامه
- **Code Editor** : برای نوشتن و ویرایش کد برنامه
- **Project Manager** : برای مدیریت بر روی فایل های پروژه

طراحی برنامه های کاربردی

با استفاده از دلفی می توانید هر نوع برنامه ۳۲ بیتی را طراحی و اجرا نمائید. از برنامه های سودمند کوچک گرفته تا برنامه های تجاری و نرم افزارهای توزیع شده با استفاده از بورلند دلفی نسخه هفت قابل توسعه و تعمیم هستند. در حین طراحی رابط کاربری برنامه توسط دلفی، در پشت پرده Form Designer کدهای مربوط به طراحی را به صورت خودکار تولید می کند و نیازی به نوشتن کدهای طراحی به صورت دستی نخواهید داشت. زمانی که شما مشخصات کامپوننت ها و اجزای بصری و غیر بصری برنامه را تغییر می دهید، Form Designer به صورت خودکار تمامی تغییرات شما را روی سورس کد برنامه اعمال میکند و تنها زحمت شما Re-Build کردن برنامه خواهد بود. شما به راحتی می توانید سورس برنامه خود را توسط هر ویرایشگر متنی (مثل Notepad) باز کرده و تغییر دهید اما بهترین ویرایشگر برای این کار ویرایشگر داخلی دلفی یا Code Editor دلفی است که با داشتن خصوصیات منحصر به فرد بهترین گزینه برای شما خواهد بود.

شما به راحتی می توانید با استفاده از زبان دلفی کامپوننت های خود را ساخته و آن ها را به قسمت Component Palette اضافه کنید. در صورت نیاز می توانید Component Palette را به دلخواه خود سفارشی نمائید. همچنین می توانید برنامه هایی تولید کنید که هم در ویندوز و هم در لینوکس قابل انتقال و اجرا باشند. این نوع از برنامه ها باید با استفاده از کتابخانه بورلند برای Cross-Platform نوشته شوند. کتابخانه CLX شامل کلاسهایی است که قادرند در دو سیستم عامل ویندوز و لینوکس بدون هیچ مشکلی کار کنند.

- **ایجاد پروژه های نرم افزاری**

هر گاه که شما در دلفی شروع به نوشتن برنامه ای می کنید در واقع در حال ایجاد یک پروژه نرم افزاری هستید. پروژه شامل تعدادی فایل مرتبط با هم می باشد که در نهایت برنامه کاربردی را تشکیل می دهند. بعضی از این فایل هاهنگام طراحی برنامه ساخته می شوند و بعضی دیگر به صورت خودکار در هنگام کامپایل برنامه.

شما می توانید محتویات پروژه تان را در پنجره Project Manager دلفی مشاهده کنید و از همان جابه مدیریت پروژه بپردازید. امکان ترکیب پروژه های مختلف در قالب یک پروژه و انجام کارهای گروهی از دیگر مزایای مدیریت پروژه در محیط دلفی است.

- **خطایابی آسان**

یکی از مباحث مهم در توسعه نرم افزارها، مبحث خطایابی است. دلفی شامل یک خطایاب (**Debugger**) داخلی است که در پیدا کردن و رفع خطاهای پروژه به شما کمک بسیاری می کند. این Debugger به شما امکانات مختلفی از جمله کنترل اجرای برنامه، مشاهده مقادیر متغیرها و داده ها و تغییر دادن آن ها را به شما می دهد. این Debugger قادر به شناسایی خطاهای هنگام اجرا Runtime Errors و برخی از خطاهای منطقی Logical Errors است.

- **توزیع برنامه کاربردی**

دلفی برخلاف Visual Basic و زبان های تحت دات نت، برنامه هایی تولید می کند که بدون هیچ پیش نیازی در سیستم عامل ویندوز اجرا می شوند. این مزیت بسیار بزرگی است که دلفی با تکیه بر آن صدها هزار توسعه دهنده را جذب خود کرده

است.

برای توزیع برنامه های دلفی روی لینوکس به Kylix نیاز خواهید داشت.

- **کدام نسخه دلفی؟**
- در این نوشته شما با خصوصیات دلفی نسخه ۷ محصول شرکت بورلند آشنا شدید. این نسخه در سال ۲۰۰۱ به بازار آمده است و تا به حال برنامه های بسیار زیادی بوسیله این محیط برای سیستم عامل ویندوز نوشته شده است. با اینکه هفت سال از توزیع این نرم افزار می گذرد اما هنوز هم برای توسعه نرم افزارهای مختلف و محصولات قدرتمند مورد استفاده قرار می گیرد. در ایران بسیاری از شرکت های کوچک و بزرگ نرم افزاری از دلفی نسخه ۷ برای تولید محصولات خود بهره می برند. البته استفاده از این محصول در جهان در حال کاهش است و حضور دلفی ۷ در توسعه برنامه های کاربردی هر روز کمرنگ تر می شود اما در ایران هنوز برای استفاده جا دارد. چون بسیاری از شرکت های دولتی و بعضاً خصوصی در ایران امکانات سخت افزاری بسیار پائینی دارند، شرکت های نرم افزاری امکان تغییر تکنولوژی و استفاده از تکنولوژی های روز دنیا همچون NET. Microsoft و JAVA را ندارند و قدرت و شجاعت این ریسک را هم ندارند.
- اما برای کسانی که همیشه دنبال جدیدترین ها هستند، نسخه ۲۰۰۷ آخرین نسخه دلفی است که شرکت CodeGear یکی از شرکت های زیر مجموعه Borland آن را تولید و توزیع کرده است. این نسخه به راحتی در بازارهای ایران پیدا می شود.

ویژگی های زبان برنامه نویسی ADA و کاربردهای آن:

Ada برای سیستم های بلادرنگ و نهفته ارائه شده است و کماکان برای همان اهداف استفاده می شود. از ویژگی های قابل توجه Ada، تایپ قوی، چک زمان اجرا، پردازش موازی، دسته بندی استثناء وgenericها می باشد Ada 95 .پشتیبانی برای برنامه نویسی شی گرا، شامل مخابره پویا و قالبهای مشابه C++ را اضافه کرده است.

پیاده سازی های Ada نوعاً از جمع آوری زباله برای مدیریت حافظه استفاده نمی کند Ada .یک شکل محدود از مدیریت حافظه بر اساس ناحیه را پشتیبانی می کند، که اجازه ی بعضی از موارد دسترسی به حافظه ی تخصیص نیافته را که باید زمان کامپایل شناسایی شود می دهد.

Adaچک های موقع اجرا را به منظور جلوگیری از دسترسی به حافظه ی تخصیص نیافته، خطاهای سرریز بافر و اشکال های قابل اجتناب دیگر انجام می دهد. این چک ها می توانند برای افزایش کارآیی از کار بیافتد Ada .همچنین شامل ابزارهایی برای کمک به تغییر برنامه است. به همین دلیل، این زبان در سیستم های بحرانی مانند ارتباطات هوایی، تسلیحات و فضاپیما استفاده ی گسترده ای دارد.

در ضمن تعداد زیادی از چک های زمان کامپایل برای کمک به جلوگیری از اشکالات را حمایت می کند که قابل شناسایی تا زمان اجرا در بعضی دیگر از زبان ها نیست، یا نیاز به چک های صریح دارد که به کد اصلی اضافه شود.

تعریف زبان Ada بین استانداردهای سازماندهی بین المللی برای استاندارد سازی در این که محتوای رایگان دارد غیر معمول است. یکی از نتایج آن این است که سند استاندارد (به عنوان کتاب"مرجع دستی" یا "RM" شناخته شده) یک مرجع معمول است که برنامه نویسان Adaبرای جزئیات تکنیکی به آن متوسل می شوند به همین ترتیب به عنوان کتاب درسی استاندارد، زبان های برنامه نویسی دیگر را آماده می کند.

اهداف طراحی زبان Ada

Ada در ابتدا برای ۳ هدف طراحی شده بود:

۱- قابلیت اطمینان و نگهداری برنامه (Program reliability and maintenance)

۲- برنامه نویسی به عنوان یک فعالیت انسانی (Programming as a human activity)

۳- کارایی (Efficiency)

این تجدید نظر زبان طراحی شده بود که انعطاف پذیری و قابلیت توسعه بیشتر ، کنترل اضافی برای مدیریت حافظه و همزمانی و پکیج های استانداردگرا در جهت پشتیبانی از نواحی کاربردی را می نماید .

از ویژگیهای زبانها این است که با قابلیت اعتماد بالا و نگهداری ساده به خوبی انتشار یابند از این رو اهمیت بر روی اعتماد برنامه ، در حالت نوشتن قرار داده شده بود. برای مثال نقش های زبان نیاز داشتند که متغیر ها به صورت صریح تعریف شوند و نوع آنها مشخص شوند . از آنجا که نوعهای یک متغیر مشابه است ، کامپایلرها می توانند از اینکه عملیات بر روی متغیر ها با خصوصیات الحاق شده برای نوع object ها سازگار است مطمئن باشند . زبانها اغلب از کامپایلهای مجزا برای واحد های مختلف برنامه پشتیبانی می کنند تا توسعه و نگهداری برنامه آسان شود واز همان درجه چک کردن داخل واحد ها برای چک کردن بین واحد ها استفاده میکند .

یک برنامه Ada ترکیب یک یا چند واحد برنامه ایست . واحد های برنامه شامل زیر برنامه ها Package ها ، Taskها و واحد های محافظت شده است .

هر واحد برنامه معمولا شامل دو بخش است

۱- Specification : شامل اطلاعاتی است که باید توسط واحد های دیگر قابل دیدن باشد

۲- Body:شامل جزیات پیاده سازی است که نیازی نیست برای دیگر واحدها قابل دیدن باشد.

یک برنامه Ada به صورت نرمال از واحد های برنامه های کتابخانه برای امکانات کلی استفاده میکند.

همه کتابخانه ها با یک روش سلسله مراتبی ساختاربندی میشوند، این تجزیه منطقی یک زیر سیستم را به مولفه های فردی اش تجزیه می نماید. متن واحد برنامه کامپایل شده مجزا باید نام واحدهای کتابخانه ای که نیاز دارد را ذکرکند.

تعریف برنامه های متن باز(Open Source)

هر وقت نامی از اوپن سورس دیده می شود، ذهن کاربران به سمت آزاد و رایگان بودن یک نرم افزار می رود، این در حالی است که نرم افزارهای اوپن سورس می توانند فروخته شوند و به صورت تجاری از آن ها استفاده شود. در حقیقت **بخش مهمی از سود صنعت نرم افزار جهان از طریق نرم افزارهای اوپن سورس تامین می شود**. سود مالی نرم افزارهای اوپن سورس در اکثر مواقع از فروش سرویس های مربوط به آن ها بدست می آید. **مدل تجاری اوپن سورس به جای تمرکز بر روی فروش خود نرم افزار، بر روی فروش خدمات مربوط به آن نرم افزار تاکید دارد.** از نگاه مشتریان جدی، یک محصول نرم افزاری بدون پشتیبانی فنی و خدمات نگهداری هیچ ارزشی ندارد. هر چقدر هم مشخصات یک محصول نرم افزاری با نیازهای یک مشتری هماهنگ باشد، باز هم مشتری نیاز به ایجاد در تغییرات در نرم افزار و هماهنگ کردن آن با شرایط خود را دارد. نرم افزار آزاد و کدباز به صورت پیش فرض هیچ تضمینی برای درست کار کردن محصول و جوابگو بودن نیازهای یک مشتری نمی دهد. این نکته به صراحت در تمام مجوزهای آزاد ذکر شده است. بنابراین اگر کاربری واقعاً به یک نرم افزار اوپن سورس رایگان نیاز اساسی داشته باشد حاضر خواهد بود برای خدمات پشتیبانی، آموزش، نگهداری و انجام تغییرات هزینه کند.

مدل تجاری Open Source یک مدل تجاری پذیرفته شده و سود آور در سطح جهان است. شرکت های معروف و تاثیرگذار زیادی در دنیای نرم افزار از این مدل برای سود آوری استفاده می کنند. از این شرکت های معروف می توان به چند نمونه اشاره کرد :

- شرکت Canonical سیستم عامل اوبونتو را به صورت رایگان عرضه می کند، در حالی که قراردادهای پشتیبانی تجاری می بندد
- Novell سیستم عامل OpenSUSE را رایگان می دهد، اما SUSE Linux Enterprise را می فروشد
- Mozilla مرورگر فایرفاکس را مجانی می دهد، و برای سودآوری قراردادهای همکاری با گوگل و شرکت های دیگر دارد
- Adobe محصول Flex خود را رایگان عرضه می کند، اما محیط توسعه آن را می فروشد
- Microsoft پلت فرم دات نت خود را رایگان می دهد، در حالی که محیط توسعه آن را می فروشد
- MySql مجانی است، اما برای داشتن پشتیبانی و قابلیت های اضافی باید پول بدهید
- Oracle نرم افزار OpenOffice را رایگان عرضه می کند، و نسخه تجاری آن یعنی StarOffice را می فروشد
- همانطور که مشاهده می کنید شرکت های بزرگ دنیای نرم افزار از این مدل برای سودآوری محصولات خود استفاده می کنند و این یعنی **مدل تجاری اوپن سورس یک مدل تجاری مطمئن و پایدار برای رسیدن به سود آوری بلندمدت در صنعت نرم افزار است.** رمز موفقیت این مدل تجاری در **وابسته کردن مشتریان به محصول نرم افزاری** است. به طور مثال بنیاد موزیلا کاربران حرفه ای وب را به مرورگر فایرفاکس وابسته کرده است و در کنار آن قراردادهای پرسودی با شرکت گوگل و دیگر شرکت ها بسته است. این کار اصلاً غیر اخلاقی نیست، موزیلا یک محصول با کیفیت را به صورت رایگان در اختیار کاربرانش قرار می دهد و در قبال آن بدون ایجاد مزاحمت خاصی برای این کاربران، درآمد کسب می کند.
- در اکثر مواقع مجوز اوپن سورس باعث پیشرفت یک محصول نرم افزاری و محبوب شدن آن میان کاربران می شود. نرم افزاری که محبوب شود، به طورگسترده مورد استفاده قرار می گیرد و این یعنی یک مارکت خوب برای سازندگان آن محصول که پتانسیل بالایی برای سودآوری دارد. پول درآوردن از راه اوپن سورس نه تنها بد نیست بلکه فواید بسیار زیادی برای صنعت نرم افزار در سطح جهان دارد. **ایجاد فرصت های شغلی یکی از فواید مهم مدل تجاری اوپن سورس است.** به طور مثال نرم افزار مدیریت محتوای WordPress را در نظر بگیرید. همه می توانند به رایگان نسخه ای از این محصول اوپن سورس را دریافت و روی هاست خود نصب کنند و وب سایت خود را مدیریت کنند. وردپرس با داشتن حجم عظیمی از پوسته ها و پلاگین های رایگان باز هم برای وب سایت های حرفه ای کافی نیست. به همین دلیل بسیاری از افراد و شرکت های کوچک به کار طراحی پوسته و پلاگین برای وردپرس مشغول هستند و از این راه پول در می آورند. سایت های نیرویابی و کاریابی پر است از شغل های مربوط به طراحی پوسته و پلاگین برای وردپرس. کافیست سری به این سایت ها بزنید. خیلی ها حاضرند برای سفارشی سازی و توسعه وردپرس بر اساس نیازهایشان به شما پول خوبی بدهند. وردپرس نمونه ای از ده ها CMS اوپن سورسی است که باعث ایجاد فرصت های شغلی بسیار زیاد برای طراحان و توسعه دهندگان وب شده است. مدل تجاری اوپن سورس به خوبی در زمینه نرم افزارهای CMS جوابگو بوده، به طوری که کمتر CMSی را پیدا خواهید کرد که موفق باشد اما اوپن سورس نباشد!

در حالی که مدل تجاری اوپن سورس یک مدل بسیار موفق و سودآور در سطح جهان است، در ایران مدلی غیر حرفه ای و مظلوم است. غیر حرفه ای به این معنی که در ایران کسی یا شرکتی به صورت حرفه ای از این مدل برای کسب درآمد استفاده نمی کند. نداشتن فرهنگ استفاده از نرم افزار، چه تجاری و چه اوپن سورس در میان مردم عامی و حتی کاربران حرفه ای کامپیوتر در ایران یکی از دلایل مهم استفاده نکردن از این مدل تجاری است. عدم آگاهی مدیران و حتی متخصصان صنعت نرم افزار ایران در مورد مدل تجاری اوپن سورس به علاوه دلیل قبل باعث شده تا استفاده از این مدل تجاری موفق در ایران کار مشکل و گاه غیر ممکنی باشد.

با فراگیر شدن وب و وب سایت های اینترنتی، ابزارهای گوناگونی برای تولید برنامه هایی مبتنی بر وب عرضه شد. یکی از ابزارهایی که در این چند سال اخیر توانست قسمت اعظمی از بازار تولید نرم افزارهای تحت وب را تسخیر کند بی شک **PHP** بوده است. PHP زبان اسکریپت نویسی ساده ای است که شامل کتابخانه ها و توابع نقریباً کاملی برای استفاده در برنامه های تحت وب است و کمتر نیازی در وب یافت می شود که با PHP نتوان آن را رفع نمود. PHP توانست در این چند سال علاقه مندان فراوانی را جذب خود کند و پروژه های بسیار قدرمتند و معروفی همچون **وردپرس** به این زبان نوشته شده اند.

PHP از نسخه های اولیه تا به حال تغییرات زیادی کرده است و فریم ورک های بسیاری برای هرچه سریع تر نوشتن برنامه های تحت وب توسط شرکت ها و افراد مختلف عرضه شده است. **یک فریم ورک PHP به شما کمک می کند تا وب سایت های خود را سریع تر و مناسب تر از همیشه بسازید.**

ویژگی های زبان برنامه نویسی PHP و کاربردهای آن:

زبان برنامه نویسی PHP یکی از محبوبترین و قدرتمند ترین زبان های متن بازدنیاست که بیشتر در موارد وب و سرور ها استفاده میشود. این زبان که نام آن مخفف کلمه hypertext preprocessor میباشد در سال ۱۹۹۴ توسط آقای لردورف ساخته شد.

بعد از ساختن ابتدایی زبان PHP توسعه دهندگان به سراغ آن آمدند و تغییرات کوچک و بزرگی را در آن اعمال کردند ، البته زبان PHP شباهت زیادی به زبان های C و پرل دارد و این موجب شده هست تا برنامه نویسان بیشتر سراغ آن بروند و محبوبیت بیشتری نسبت به دیگر زبان ها دارد.

بعد از انتشار نسخه اول یک سال بعد ورژن ۲ پی اچ پی منتشر شد و ورژن ۳ آن ۲ سال بعد و ورژن ۴ سال ۲۰۰۰ ودر آخر ورژن ۵ آن ۴ سال بعد یعنی ۲۰۰۴ منتشر شد و تا به حال ورژن جدید دیگری منتشر نشده است

php۵که آخرین ورژن این برنامه است که دارای ویژگی‌هایی مانند پشتیبانی از زبان شی گرا (برای پایگاه داده) ووسعت بسیار زیاد کارایی آن است.

البته به گفته ی خود سازندگان کاربران ویژگی های بسیار مهمی را در ورژن ۶ خواهند دید.

این زبان برنامه نویسی خدمت زیادی به کاربران اینترنت کرده است و از زمانی که این زبان متولد شد در جامعه مجازی تغییرات زیادی انجام شد ، همچنین نرم افزار های تحت وب بسیاری ساخته شد که محبوبترین و مشهور ترین آنها MyBB و مامبو هست که شما هم میتوانید آنها را به رایگان دانلود کنید و روی سایت خود نصب کنید.

همانطور که در قبل گفتیم این زبان شباهت زیادی به زبان C دارد ولی از نسخه ۵ به بعد شباهت آن بیشتر به زبان جاوا اسکریپت کشیده شده هست و باز محبوبیت بیشتری در بین برنامه نویسان پیدا کرد و این باعث شده تا برنامه نویسان برنامه های قدرتمند خود را در کوتاه ترین زمان ممکن طراحی کنند.

برخی از ویژگی های این زبان:

*امکان تغییر نوع کاربردی از اسکریپت نویسی به گرافیک

*امکان استفاده و اتصال به انواع پایگاه ها مانند MySQL

*امکان اجرا بر روی اکثر سیستم عامل ها مانند ویندوز و لینوکس

برخی از فرم ورک های PHP

Zend Framework: این فریم ورک که از طرف سازندگان PHP عرضه شده است، یکی از کاملترین فریم ورک های PHP است. دارای توابعی برای تامین امنیت وب سایت های شما و همچنین توابعی برای استفاده از سرویس های وب ۲.۰ مثل سرویس های گوگل، یاهو، آمازون و فلیکر است. این فریم ورک بر روی ساخت وب سایت های کاربردی و وب سرویس ها با رویکرد وب ۲.۰ تمرکز دارد.

CakePHP: یکی از قدرتمند ترین فریم ورک های PHP با روش استفاده آسان و پشتیبانی از مدل های طراحی MVC و ORM. این فریم ورک زمان توسعه و کدنویسی را به حداقل ممکن می رساند و کمک می کند تا برنامه های تحت وب قدرتمندتری بسازید.

Qcodo: فریم ورک اوپن سورس برای PHP 5 که به شما در ساخت وب سایت های کاربردی کمک بسیاری می کند. توسعه دهندگان به جای اینکه هفته ها وقت خود را صرف کارهای تکراری بکنند، می توانند به سادگی از توابع از پیش تعریف شده در این فریم ورک بهره

ببرند. این فریم ورک کاملاً بر اساس برنامه نویسی شیء گرا پیاده سازی شده است و پلت فرمی برای ایجاد سریع برنامه های تحت وب فراهم می کند.

Symfony : یک فریم ورک قدرتمند دارای کلاس های متعدد که ساختن وب سایت های پیچیده را آسان تر می کند. انتخاب این فریم ورک به شما کمک می کند تا بدون دردسر برنامه های تحت وب خود را توسعه دهید و آن ها را زودتر از انتظار بقیه آماده کنید. بسیاری از توسعه دهندگان PHP از این فریم ورک استفاده می کنند.

Seagull : فریم ورک اوپن سورس که با مجوز BSD توزیع شده است. به توسعه دهندگان PHP امکانات و ابزارهای مختلفی ارائه می کند تا برنامه های خود را هر چه سریع تر و کاراتر تولید کنند. همچنین این فریم ورک قابلیت های خوبی برای توزیع برنامه های PHP روی وب یا به صورت محلی (Local host) در اختیار برنامه نویسان قرار می دهد و دارای اجتماع کاربری مناسبی است.

Solar : فریم ورکی برای نوشتن سریع وب سایت هایی است که با PHP 5 ساخته می شوند. این فریم ورک از مدل های طراحی تجاری پشتیبانی می کند و به صورت توکار از بومی سازی (Localization) پشتیبانی می کند.

Prado : فریم ورکی است بر اساس کامپوننت که شما را قادر به ساختن برنامه های تحت وب به زبان PHP با قابلیت های برنامه نویسی شیء گرا می کند.

Codeigniter : یکی از قدرتمند ترین فریم ورک های PHP است که به توسعه دهندگان امکان ساخت وب سایت هایی با کارایی بالا را می دهد. این فریم ورک دارای آموزش های مناسب و مستندات بسیار است تا شما هر چه آسان تر نحوه ی کار با آن را یاد بگیرید.

AjaxAC : فریم ورک اوپن سورس برای استفاده از تکنولوژی Ajax در برنامه های مبتنی بر PHP است.

xAjax : کتابخانه ای از کلاس ها برای استفاده از تکنولوژی Ajax در زبان PHP است. با این کتابخانه کلاس می توانید با استفاده از جاوا اسکریپت، CSS، HTML و PHP برنامه های تحت وب قدرتمند و با بازدهی بالا بنویسید.

دیگر فریم ورک های PHP : فریم ورک های بسیاری تا به حال برای PHP نوشته شده اند. همه آن ها سعی کرده اند با فراهم آوردن توابع و کلاس های لازم، زمان توسعه را برای برنامه نویسان به حداقل برسانند. شما بایستی با توجه به نیازهای خود یکی از این فریم ورک ها را انتخاب کرده و سپس نحوه ی کار با آن را یاد بگیرید. پس از مدتی که از این فریم ورک ها در برنامه های تحت وب خود استفاده کنید، متوجه کارایی و صرفه جویی در زمان و هزینه خواهید شد.

ویژگی های زبان برنامه نویسی پیتون و کاربردهای آن:

پایتون یک زبان برنامه‌نویسی سطح بالا، شیءگرا و تفسیری است که توسط گیدو ون روسوم (Guido van Rossum) در سال ۱۹۹۰ طراحی شد.

این زبان از زبان های برنامه نویسی تفسیری بوده و به صورت کامل یک زبان شی‌گرا است که در ویژگی‌ها با زبانهای تفسیری پرل، روبی، اسکیم، اسمال‌تاک و تی‌سی‌ال مشابهت دارد و از مدیریت خودکار حافظه استفاده می‌کند.

پایتون پروژه‌ای بازمتن توسعه یافته است و توسط بنیاد نرم‌افزار پایتون مدیریت می‌گردد. نسخهٔ کنونی (مارس ۲۰۰۸) این زبان ۲.۵.۲ است.

پایتون در یک محیط آموزشی ایجاد و توسعه یافته است . یعنی در کریسمس سال ۱۹۹۸ در موسسه ملی تحقیقات ریاضی و رایانه (CWI) در شهر آمستردام. در آن زمان گیدو یک پژوهشگر در CWI بود و در زمان بیکاری خود بر روی پروژه شخصی خود یعنی پایتون کار می کرد . اولین نسخه عمومی از پایتون در ماه فوریه سال ۱۹۹۱ منتشر شد . برای مدتی نسبتاً طولانی پایتون توسط موسسه ملی تحقیقات و ابتکارات (CNRI) واقع در رستون ایالات متحده امریکا توسعه می‌یافت . تا اینکه در سال ۲۰۰۰ تیم توسعه دهنده پایتون به آزمایشگاه های پایتون منتقل شدند . نام پایتون از برنامه مورد علاقه سازنده آن یعنی مونتی پایتون که یک برنامه کمدی انگلیس بود گرفته شده است.

پایتون یک زبان برنامه‌نویسی شی‌گرا است و از ویژگی های پیشرفته‌ایی چون وراثت، چند شکلی، سربار گزاری عملگر و ... پشتیبانی می کند. یک از ویژگی‌های پایتون که لقب چسب را برای پایتون به ارمغان آورده امکان استفاده از کد ها و کلاس‌های نوشته شده در زبانهای دیگری چون سی‌پلاس‌پلاس و جاوا است که در حقیقت کار چسباندن قطعات کد جدا و فقط نوشتن بدنه اصلی به عهده پایتون است .

پایتون یک زبان برنامه‌نویسی رایگان و متن‌باز(open source) هست .می‌توانید متن آن و خود برنامه را به رایگان از اینترنت دریافت یا در توسعه آن همکاری کنید.

چون پایتون با زبان قابل حمل سی نوشته شده می تواند به صورت مجازی بر روی هر پردازشگری همگردانی و اجرا شود . ماشین مجازی (مفسروبندوز بنویسید و سپس بدون تغییر روی لینوکس یا مکینتاش یا هر سیستم عامل و سخت‌افزار دیگری که پایتون روی آن نصب باشد اجرا کنید .

پایتون زبانی چند رگه است که از زبان‌های برنامه‌نویسی تفسیری (برای مثال :تی‌سی‌ال، اسکیم، پرل) و زبان‌های سیستمی (برای مثال: سی‌پلاس‌پلاس، سی و جاوا) مشتق شده. بنابراین تمام سادگی و راحتی کار زبان‌های برنامه‌نویسی تفسیری و ویژگی‌ها و قدرت زباهای سطح پایین را داراست .

شما می توانید قطعه از کد را در زبانی چون سی‌پلاس‌پلاس، سی و جاوا در پایتون استفاده کنید. و یا می توان از توابع کتابخانه‌ای و کامپوننت هایی چون COM API استفاده کرد .البته نوع این نوع برنامه نویسی (ماژول) با برنامه نویسی معمولی هر زبان متفاوت می باشد . می توان از کد های پایتون در زبانهای دیگر نیز استفاده کرد (درونی سازی)

مقایسه زبان PHP و پیتون:

لازم به ذکر است زبان php جزو زبان های پرکاربرد و قدرتمند در زمینه وب بوده و زبان پایتون نیز علاوه بر کاربردهای فراوان دیگر آن (ساخت بازی های کامپیوتری گرفته تا نرم افزارهای موبایل و نرم افزارهای دسکتاپ برای لینوکس و ویندوز و ...) در زمینه وب نیز علیرغم تازه وارد بودن (خصوصا در ایران) بسیار قدرتمند ظاهر شده و پا به پای دیگران در این عرصه رقابت می کند. زبان پایتون توسط ابرقدرت هایی همچون ناسا، گوگل و ... به طور گسترده ای مورد استفاده قرار گرفته است.

زمینه های مشترک

هر دو زبان python و php دارای ویژگی های زیر می باشند:

*زبان های تفسیری، سطح بالا و دارای انواع پویا می باشند

*متن باز هستند (مگر زمانی که انواع محصولات zend برای استفاده از php توصیه می شوند)

*دارای پشتیبانی توسط جوامع بزرگ توسعه دهنده

*یادگیری آنها آسان است (در مقایسه با java وperl)

*امکان توسعه آسان در c ، c++و جاوا

*قابلیت حمل بالا دارند. این زبانها بر روی اکثر پلتفرم ها بدون کامپایل مجدد قابل اجرا می باشند.

*پشتیبانی از تعداد متغییر آرگومان توابع

*قابلیت آزادسازی آبجکت های فعال در ارائه رشته

مقایسه دو زبان

ویژگی هایی که php دارد و پایتون ندارد:

*دستور زبان مشابه C و Perl ، همراه با تعداد زیادی علامت دلار و جفت گیومه

*عبارت 'switch' و حلقه'do ... while'

*عملگرهای تخصیص و افزایش کاهش (تخصیص در پایتون فقط عبارت می باشد)

*عملگر/عبارت سه تایی(... : ... ? ...)

*تابلو اسکیزوفرنیک از نام تابع

*محیط مصلحتی (عموما نصب شده)

*تغییر خصوصی، محفوظ و عمومی هم برای متدها و هم خواص

*تغییر انتزاعی و نهایی هم برای کلاس ها و هم متدها

*اینترفیس ها

ویزگی هایی که Python دارد و php ندارد:

*زبان برنامه نویسی چندکاره (نه فقط برای وب)

*برای نمایش ساختار بلوک به جای استفاده از کروشه از تورفتگی خطوط استفاده می شود

*فضاهای نام و ماژول ها

*هسته کوچک

*دستور زبان بسیار روشن، مختصر و متعامد

*زبانی خود مستند توسط docstrings وpydoc

*آرگومان های کلیدواژه برای توابع و متدها، پشتیبانی آسان از آرگومان های پیش فرض

*شی گرایی صحیح و توابع و کلاس های 'first class' صحیح

*کلاس ها به صورت گسترده ای در کتابخانه های استاندارد استفاده شده است

*مفهوم صفات خصوصی

*وراثت چندگانه

*مدیریت فایل شی گرا

*زنجیری متد

*خودکاوی عالی

*همه چیز مرجع است! (مراجع در PHP رنج آور هستند)

*یک عبارت 'del' برای همه ی انواع داده. (در PHP از 'unset' برای متغییرها و چیز دیگری برای اعضای آرایه استفاده می شود)

*case sensitivity سازگار (توابعPHP ، حساس به حروف نیستند ولی متغیرها هستند)

*دستور زبان برش آرایه ساده

* lambdas و دیگر ساختارهای برنامه نویسی تابعیbuiltin

*تکرار کننده ها

*ساختار پردازش استثنا

*سربارگذاری عملگر

*یکپارچه سازیSWIG

*threading

*profiler عالی

*چندین debugger وIDE

* *پشتیبانی از همه ی فریم ورک های GUI بزرگ

*بین المللی سازی قدرتمند و پشتیبانی از پونیکد

*جاافتادگی، ثبات و سازگاری صعودی

*تمایل به رهبری برنامه های مقیاس پذیر خیلی بیشتر –– وارد کردن ماژول ها ایمن تر از ضمیمه کردن کدها ازروی متن در php می باشد: متغییرهای عمومی به منظور تبادل اطلاعات استفاده نمی شوند

ویژگی های زبان برنامه نویسی java و کاربرد آن:

جاوا یک زبان ساده ، شی گرا ،توزیع شده ، تفسیر شده ، قدرتمند ، ایمن ، با معماری خنثی ، قابل حمل ، با عملکرد سطح بالا چند نخ کشی شده و پویا است .

Sunتصدیق میکند که به طور قطع این کلمات رشته هایی از واژه های متداول در زبان برنامه نویسی هستند ، اما حقیقت این است که این واژه ها به طور ماهرانه ای خصوصیات این زبان را شرح می دهند .

حال به برخی از خصلت های جاوا در پشت این واژه ها میپردازیم .

شی گرا Object Oriented:

جاوا یک زبان برنامه نویسی شی گرا است . برای یک برنامه نویس این به این معنا است که به جای فکر کردن به قسمت های رویه برنامه ، باید به کاربرد داده ها و روش هایی که روی آن داده ها عمل میکنند ، توجه شود .

اگر شما به برنامه نویسی با اعلان رویه در C عادت کرده اید ،ممکن است دریابید که به هنگام استفاده از جاوا مجبور به تغییر در روش و چگونگی برنامه تان هستید . هنگامی که فهمیدید این الگوی جدید چقدر قدرتمند است ، به سرعت با آن هماهنگ میشوید .

در یک سیستم شی گرا ، یک کلاس مجموعه ای از داده ها و روش هایی است که روی آن داده عمل میکنند. همراه بودن داده ها و متد ها رفتار و حالت یک شی را بیان می دارد . کلاس ها به صورت سلسله مراتبی مرتب شده اند ، بنابر این یک زیر کلاس میتواند رفتار هایی را از کلاس بالاتر به ارث ببرد . یک کلاس سلسله مراتبی همیشه یک کلاس ریشه دارد که کلاسی است با رفتار های کاملا عمومی .

جاوا به همراه دسته ی گسترده ای از کلاس هایی است که در بسته هایی مرتب شده اند و شما می توانید از آنها در برنامه ی خود استفاده کنید . برای مثال جاوا کلاس هایی را ایجاد میکند که :

بخش های رابط گرافیکی را می سازند (the java.awt package) ،کلاس هایی که عملیات ورودی و خروجی را به عهده دارند (the

(java.io packageو کلاس هایی که از شبکه پشتیبانی میکنند . (the java.net package)

یک شی کلاس (in the java.lang package)به عنوان ریشه کلاس سلسله مراتبی جاوا انجام وظیفه میکند .

جاوا بر خلاف C++ طوری طراحی شده است که از همان ابتدا به صورت شی گرا باشد . اکثر چیز ها در جاوا اشیا هستند . ارقام ابتدایی ، کاراکترها و مدل های منطقی تنها استثناء ها هستند . حتی رشته ها هم در جاوا به وسیله اشیا حاضر میشوند ،همان طور که ساختمان های مهم دیگر این زبان ، مثل نخ ها احضار میشوند . یک کلاس یک واحد پایه برای کامپایل و اجرا شدن در جاوا است .تمام برنامه های جاوا متشکل از کلاس ها است .

درست است که جاوا طوری طراحی شده است که مثل C++ باشد و خاصیت های آن را داشته باشد ، اما هنگامی که با آن کار کنید خواهید فهمید که بسیاری از پیچیده گی های آن زبان را از بین برده است .

اگر شما یک برنامه نویس C++ هستید حتما لازم است که ساختار های شی گرایی در جاوا را به دقت مطالعه کنید . اگرچه ترکیب و نحوه دستورات آن تقریبا شبیه C++ است ،اما رفتار های آن خیلی مشابه نیست .

تفسیر شده Interpreted:

جاوا یک زبان تفسیر شده است . کامپایلر جاوا به جای ایجاد کد محلی ماشین ، کد بایتی برای ماشین مجازی جاوا ایجاد میکند . برای اجرای دقیق برنامه ، از مفسر جاوا برای اجرای کد های بایتی کامپایل شده استفاده میشود .به دلیل اینکه کد های بایتی جاوا به نوع کامپیوتر بستگی ندارند ، برنامه های جاوا میتوانند روی هر نوع کامپیوتری که JVM (Java Virtual Machine) را دارند ، اجرا شوند . در محیط تفسیر شده ، مرحله لینک استاندارد توسعه برنامه از دید کاربر پنهان است . اگر جاوا تنها یک مرحله لینک داشت ، فقط بارگذاری کلاس جدید به محیط پردازش می شد که یک پردازش نموی سبک وزن است که در زمان اجرا مشاهده میشود . که این خصوصیت با چرخه کامپایل–لینک–اجرا ی آرام و طاقت فرسای زبان هایی مانند C یا C++ در تضاد است .

معماری خنثی و قابل حمل Architecture Neutral and Portable:

به دلیل اینکه برنامه های جاوا در فرمت کد بایتی با معماری خنثی کامپایل شده اند ، برنامه کاربردی جاوا میتواند در هر سیستمی اجرا شود .

البته با این شرط که آن سیستم توانایی پیاده سازی ماشین مجازی جاوا را داشته باشد . این مسئله تقریبا برای کاربرد های توزیع سده روی اینترنت و یا دیگر شبکه های ناهمگن مهم است . اما روش معماری خنثی برای کاربرد های بر مبنای شبکه مفید است .

به عنوان یک توسعه دهنده برنامه های کاربردی در بازار نرم افزاری امروز ممکن است بخواهید مدل های کاربردی خود را توسعه دهید ، به طوری که بتواند روی Pc ، مکینتاش و سیستم عامل Unix اجرا شود .با وجود گونه های مختلف Unix ، Windowsروی Pc و مکینتاش قوی جدید ، رفته رفته تولید نرم افزار برای همه انواع این کامپیوتر ها سخت می شود . اگر شما برنامه تان را در جاوا بنویسید میتواند روی همه ی این کامپیوترها اجرا شود .

در حقیقت تفسیر شده بودن جاوا وتعریف یک استاندارد ، معماری خنثی داشتن و فورمت کد بایتی آن از بزرگترین دلایل قابل حمل بودن آن به شمار می آیند .

اما جاوا باز از این هم بیشتر گام برمیدارد ، با اطمینان حاصل کردن از اینکه هیچیک از جنبه های وابستگی اجرایی زبان را ندارد . برای مثال جاوا به طور صریح اندازه هریک از انواع داده را تعریف میکند که این با C تفاوت دارد ، برای مثال هریک از انواع صحیح می تواند بسته به نوع کامپیوتر۱۶-۳۲ یا ۶۴ بیت طول داشته باشد .

هنگامی که به صورت تکنیکی امکان نوشتن برنامه های غیر قابل حمل در جاوا فراهم شد ، جلوگیری از چند خاصیت وابسته به نوع کامپیوتر که توسط جاوا API تولید شده و به طور قطع قابل حمل نوشته شده است ، آسان است .

یک برنامه جاوا به تولید کنندگان نرم افزار کمک میکند تا از قابل حمل بودن کد هایشان اطمینان حاصل کنند . برنامه نویسان فقط برای پرهیز از دام غیر قابل حمل بودن برنامه احتیاج به یک تلاش ساده دارند که شعار تجارتی شرکت Sunرا زنده نگهدارند و آن شعار این است :

»یک بار بنویس ، همه جا اجرا کن . «

پویا و توزیع شده Dynamic and Distributed:

جاوا یک زبان پویا است . هر کلاس جاوا میتواند در هر زمانی روی مفسر جاوا بارگذاری شود . سپس این کلاس های بارگذاری شده ی پویا میتوانند به صورت پویا معرفی شوند . حتی کتابخانه کد های محلی میتواند به طور پویا بارگذاری شود .کلاس ها در جاوا با کلاس Classفراخوانی میشوند ؛ شما میتوانید به طور پویا در مورد یک کلاس در زمان اجرا اطلاعاتی بدست بیاورید .این خصوصیت در جاوا ۱-۱ به طور درستی موجود است . با وجود بازتاب API اضافه شده (Application Program Interface) که به برنامه ساز امکان میدهد که با برنامه از طریق یک برنامه کاربردی دیگر ارتباط برقرار کند .

جاوا حتی با نام زبان توزیع شده نیز خوانده میشود . به طور ساده این به این معنا است که این زبان پشتیبانی سطح بالایی برای شبکه به وجود می آورد . برای مثال کلاس URL و کلاس های مرتبط با آن در بسته ی Java.net ، خواندن فایل های دوردست را به همان سادگی خواندن فایل های محلی کرده است . به طور مشابه در جاوا ۱-۱ ، احضار روش کنترلی RMI (Remote Method Invocation)، APIبه یک برنامه جاوا اجازه میدهد که روش هایی از اشیاء دور دست جاوا را به همان صورتی که اگر آن اشیاء محلی بدند آنها را میخواند ، بخواند .(جاوا حتی از سیستم شبکه ای سطح پایین که شامل آدرس مقصد و مسیر جریانی که توسط سوکت ها متصل شده است ، نیز پشتیبانی میکند.)

طبیعت توزیع شده ی جاوا زمانیکه با امکانات پویای بارگذاری کلاس همراه میشود ، واقعا درخشنده است . این خصوصیات با هم این امکان را برای مفسر جاوا به وجود می آورند که کد ها را از اینترنت بارگذاری و اجرا کند. این چیزی است که در هنگام بارگذاری و اجرای یک برنامه کاربردی از اینترنت توسط مرورگر وب ، اتفاق می افتد . اما داستان پیچیده تر از این هم میتواند باشد . تصور کنید یک پردازشگر کلمه چند رسانه ای در جاوا نوشته شده است . وقتی از این برنامه پرسیده میشود که چند نوع از داده هایی را که قبلا هرگز وارد

نشده را نمایش دهد ، ممکن است به طور دینامیکی یک کلاس را که میتواند داده را شناسایی کند ، از شبکه بارگذاری کند و بعد کلاس دیگری را که بتواند داده را از درون یک پوشه ترکیبی بخواند ، باز به طور دینامیکی بارگذاری میکند . برنامه ای مانند این از منابع توزیع شده در شبکه برای رشد و سازگاری خودکار کاربران استفاده میکند .

ساده Simple:

جاوا یک زبان ساده است . طراحان جاوا سعی در این داشتند تا زبانی بوجود بیاورند که برنامه نویسان بتوانند به سرعت آن را یاد بگیرند . بنابراین تعداد ساختار های این زبان تقریبا کم است . هدف دیگر طراحی این زبان این بود که به منظور راحتی انتقال آن ، آن را طوری طراحی کنند که برای عده ی زیادی از برنامه نویسان آشنا باشد . اگر شما یک برنامه نویس C یا C++ هستید ، خواهید فهمید که جاوا از بسیاری از ساختار های C و C++ استفاده میکند .

برای اینکه این زبان را هم به طور ساده و هم آشنا و ملموس و هم کوچک نگه دارند بسیاری از خصوصیات C و C++ را در آن حذف کردند . اینها خصوصیاتی بودند که باعث می شدند برنامه نویسی ضعیفی صورت بگیرد یا آنهایی بودند که به ندرت در برنامه استفاده می شدند . برای مثال جاوا از دستور goto استفاده نمی کند ، در عوض از دستورهای continue , break در مواقع نیاز استفاده می کند . جاوا از سر فایل ها (header files)استفاده نمی کند و پردازشگر C را هم حذف کرده است . به این دلیل که جاوا یک زبان شی گرا است ، ساختار های C مثل struct , union از آن برداشته شده است . جاوا حتی بارگذاری مجدد و خواص چندگانه ارث بری از C++ را هم حذف کرده است . شاید مهمترین پارامتر ساده بودن جاوا عدم استفاده این زبان از اشاره گر ها باشد . اشاره گر ها یکی از بیشترین موجودیت های دردسرساز در C++ , C هستند . چون جاوا ساختمان ندارد و آرایه ها و رشته ها اشیاء آن هستند ، بنابراین احتیاجی به اشاره گر نیست .جاوا به طور خودکار آدرس دهی و دسترسی به محتوای موجود در یک آدرس را برای شما انجام میدهد .

جاوا حتی زباله های حافظه ای را هم به طور خودکار جمع آوری میکند (Garbage Collectin).جمع آوری آشغال فرایندی است برای ترمیم خودکار حافظه انباشته شده . بلوک هایی از حافظه که زمانی به فایل ها اختصاص داشتند اما مدتی است که از آنها استفاده نمی شود و بلوک هایی از حافظه که هنوز مورد استفاده قرار میگیرند ممکن است حرکت داده شوند تا از به هم پیوستن فضاهای خالی حافظه بلوک های خالی بزرگتری بدست آید .

قدرتمند Robust:

جاوا برای نوشتن نرم افزارهای قدرتمند و بسیار ایمن ساخته شده است . جاوا هنوز هم به طور قطع نرم افزار ها را تضمین نمیکند . تقریبا هنوز هم امکان نوشتن برنامه های مشکل ساز در جاوا وجود دارد ، هرچند که جاوا برخی از انواع مشخص خطاهای برنامه نویسی را حذف کرده که به طرز چشمگیری نوشتن نرم افزار های ایمن را آسان تر کرده است .

جاوا یک زبان تایپ شده قدرتمند است ، که اجازه چک شدن مشکلات و خطاهای تایپی را در زمان کامپایل می دهد . جاوا بسیار قویتر از C++تایپ شده است که بسیاری از خصوصیات انعطاف پذیر در زمان کامپایل را از C به ارث برده است . مخصوصا هنگام اعلان توابع

.جاوا به مدل اعلان صریح احتیاج دارد ، زیرا که از مدل اعلان صریح C پشتیبانی نمیکند . این مسئله مارا از اینکه کامپایلر میتواند خطاهای زمان اعلان را بدست آورد ، مطمئن میکند . مسئله ای که منجر به ایجاد برنامه های ایمن تری میشود .

یکی از چیزهایی که باعث شده که جاوا ساده باشد عدم وجود اشاره گر ها ومحاسبات بر روی آنها است . این ویژگی حتی قدرت جاوا را هم با از میان بردن یک کلاس سراسری اشاره گر افزایش میدهد .

به طور مشابه تمام دسترسی به آرایه ها و رشته ها در زمان اجرا چک می شوند تا از قطعی بودن آنها اطمینان حاصل شود .با از بین بردن امکان دوباره نویسی حافظه و داده های هرزه ، تعویض نقش اشیاء از نوعی به نوع دیگر هم در زمان اجرا کنترل میشود تا از مجاز بودن آن اطمینان حاصل شود .

سرانجام زباله جمع کن خودکار جاوا بسیاری از عملیات پاکسازی مرتبط با معماری حافظه را راه اندازی میکند . چیزی که از خطاهای خطر ساز مرتبط با تخصیص و آزاد سازی حافظه جلوگیری میکند .

ایمن Secure :

یکی از دلایل پرطرفدار بودن جاوا این است که یک زبان ایمن است . این ویژگی مخصوصا به خاطر طبیعت توزیع شده ی آن بسیار مهم است .بدون وجود امنیت شما قطعا نمیخواهید که یک کد را از یک سایت تصادفی اینترنت بارگذاری کنید و به آن اجازه اجرا شدن روی کامپیوتر خودتان را هم بدهید . این دقیقا همان چیزی است که مردم هرروز با یک کد جاوا انجام میدهند . جاوا به صورت ایمن طراحی شده و چندین لایه کنترل امنیت به وجود می آورد که شما را در برابر کد های خطرساز محافظت می کنند و به کاربر اجازه میدهد که برنامه های ناشناخته را با خیال راحت اجرا کند .

همان طور که دیدیم ، برنامه جاوا نمیتواند اشاره گر ها را به حافظه یا آرایه های سرریز یا حافظه خواندنی خارج از محدوده یک آرایه یا رشته اشاره ، اشاره دهد. این خصوصیت یکی از اصلی ترین وسایل دفاع جاوا در برابر کدهای خطرساز است . دومین راه دفاع در برابر کد های خطرساز ، پردازش کدهای بایتی به صورت قابل تصدیق و تایید است که مفسر جاوا به روی هر کدی که در حال بار گذاری باشد اعمال میکند . این مراحل تایید از اینکه کد به صورت درستی ساخته شده اطمینان حاصل میکنند ، که برای مثال پشته سرریزی یا زیرریزی نداشته باشد ، یا شامل کد های بایتی غیر مجاز نباشد .

کدهای بایتی خراب یا خطرساز ممکن است از ضعف های اجرایی در مفسر جاوا سوء استفاده کنند . لایه ای که در اینجا ما را به طور ایمن محافظت میکند ، مدل جعبه شنی (Sand box) است : کدهای ناشناخته در یک جعبه شنی قرار میگیرند جایی که میتوانند به صورت ایمن اجرا شوند ، بدون اینکه هیچ صدمه ای به بقیه اجزاء یا محیط جاوا بزنند .

وقتی یک برنامه کاربردی یا دیگر کدهای ناشناخته در جعبه شنی در حال اجرا است ، چند محدودیت در مورد کاری که میتواند انجام دهد ، وجود دارد . واضح ترین این محدودیت ها این است که هیچ دسترسی به هیچ یک از فایل های محلی سیستم وجود ندارد . در جعبه شنی محدودیت دیگری هم وجود دارد که به وسیله کلاس مدیریت امنیت اعمال میشود . این مدل در ابتدا از اینکه سیستم های امنیتی را نصب کرده اید یا نه ، مطمئن میشود ، چرا که همه کلاس های جاوا نیاز به عملیات حساسی مانند دسترسی به سیستم فایل را دارند . اگر

فراخوانی به وسیله یک کد ناشناخته به صورت مستقیم یا غیر مستقیم انجام شد ، مدیر امنیت مورد استثناء را می فرستد و عملیات صورت نمی گیرد .

و سرانجام در جاوا ۱-۱ یک راه حل ممکن دیگر برای مشکلات امنیتی وجود دارد ، به وسیله ضمیمه کردن یک امضاء دیجیتالی به کد جاوا که اصل آن کد میتواند به صورت پنهانی و نهفته ساخته شود . اگر شما اعتماد خود را به یک شخص یا یک سازمان مشخص کرده باشید ، کدی که امضاء آن هویت مورد اعتماد روی آن قرار دارد ، ایمن و مطمئن است . حتی زمانیکه در حال بارگذاری شدن در شبکه است و ممکن است حتی بدون جلوگیری توسط جعبه شنی اجرا شود .

عملکرد سطح بالا High Performance:

جاوا یک زبان تفسیر شده است ، بنابر این هرگز به سرعت زبان کامپایل شده ای مثل C نخواهد بود . گفته میشود که جاوا ۱-۰ به اندازه ۲۰ برابر از Cکند تر است . جاوا ۱-۱ تقریبا سرعتی دو برابر جاوا ۱-۱ دارد . بنابراین ممکن است عاقلانه باشد اینکه بگوییم کد C کامپایل شده ۱۰ برابر سریع تر از کد های تفسیر شده ی جاوا اجرا میشود . اما قبل از اینکه به خاطر این موضوع مایوس شوید ، آگاه باشید که این سرعت بیشتر از آن چیزی است که برای برنامه های پرسرعت ، کاربردهای (GUI (Graphical User Interface ، برنامه های برمبنای شبکه ، جایی که برنامه کاربردی معمولا آماده برای اجرا شدن است ، انتظار برای کاربر که یک دستوری اعمال کند و یا انتظار برای دریافت از شبکه ، لازم است .

به علاوه قسمت هایی که به سرعت های بالا نیاز دارند ، که کارهایی از قبیل الحاق رشته ها و مقایسه را انجام میدهند ، با کد محلی جاوا اجرا میشوند .

علاوه بر این کارایی ، بسیاری از مفسر های جاوا اکنون شامل کامپایلر های فقط در زمان " just in time " نیز هستند که میتواند کدهای بایتی جاوا را برای هر نوع CPUدر زمان اجرا به کد ماشین ترجمه کند . فرمت کد بایتی جاوا با این کامپایلر های در زمان در مرکز کافی و مناسب است و انصافا کدهای خوبی تولید میکند . در حقیقت Sun ادعا میکند که کارایی کد های بایتی که به کد ماشین تبدیل شده اند ، تقریبا به خوبی کارآیی آن در C و C++ است .

اگر شما خواهان این هستید که قابل حمل بودن کد ها را قربانی بهبود در سرعت آن کنید ، میتوانید بخش قابل توجهی از برنامه خود را در Cو C++ بنویسید و از روش های مخصوص جاوا برای مشترک کردن با این کد محلی جاوا استفاده کنید .

چند نخ کشی شده Multitbreaded:

در یک برنامه کاربردی بر مبنای GUI شبکه ای ، مثل مرورگر وب ، تصور اینکه چند چیز بتوانند به طور همزمان اجرا شوند ، آسان است . یک کاربر میتواند همزمان با اینکه دارد یک صفحه وب را میخواند به یک کلیپ صوتی گوش دهد و همزمان در پس زمینه مرورگر یک عکس را بارگذاری کند .

جاوا یک زبان چندنخ کشی شده است ، که از چندین رشته اجرایی (گاهی پردازش سبک وزن خوانده میشود) پشتیبانی میکند و میتواند

چندین کار را انجام دهد . یکی از مزیت های چندنخ کشی شده این است که عملکرد سطح بالایی برای کاربردهای گرافیکی برای کاربر فراهم میکند .

اگر شما سعی کرده اید که با نخ ها در C و C++ کار کنید ، میدانید که کمی مشکل است . جاوا برنامه نویسی با نخ ها را بسیار آسان تر کرده است ، با به وجود آوردن زبان درون ساخته شده ای که از نخ ها پشتیبانی میکند . بسته jana.langیک کلاس بوجود آورده است که از روش هایی برای شروع و پایان یک نخ ، و مرتب کردن ترتیب گره ها در میان چیز های دیگر ، پشتیبانی میکند .

حتی دستورات زبان جاوا از نخ ها پشتیبانی میکنند ، که با استفاده از کلمات کلیدی مطابق شده . این کلمات کلیدی علامت گذاری بخش های کد یا تمامی روش هایی را که باید فقط با یک نخ در یک زمان اجرا شوند را به شدت آسان کرده است .

به دلیل اینکه جاوا استفاده از نخ ها را بسیار ساده میکند ، کلاس جاوا در شماری از جاها از این نخ ها استفاده میکند . برای مثال هر برنامه کاربردی که انیمیشن اجرا میکند ، از نخ ها استفاده کرده است .

به طور مشابه جاوا از برنامه های نا همگام ، ورودی ها و خروجی های بلاک نشده با اخطاری به وسیله سیگنال ها یا وقفه ها پشتیبانی نمیکند ، در این صورت شما باید یک نخ بسازید که روی هر کانال ورودی خروجی که با آن کار میکنید بلاک شده باشد.

پلت فرم دات نت و ویژگی های آن:

فریمورک دات نت ، فونداسیون لازم بمنظور ایجاد و اجرای برنامه ها و سرویس های وب XML را ارائه می نماید. ماهیت یکسان (یکدست) فریمورک دات نت ، امکان ارتباط تمامی برنامه ها(برنامه های ویندوز ، وب ، سرویس های وب XML) را از طریق یک مجموعه از ابزارها و کدهای نوشته ، فراهم می نماید. فریمورک دات نت شامل موارد زیر است :

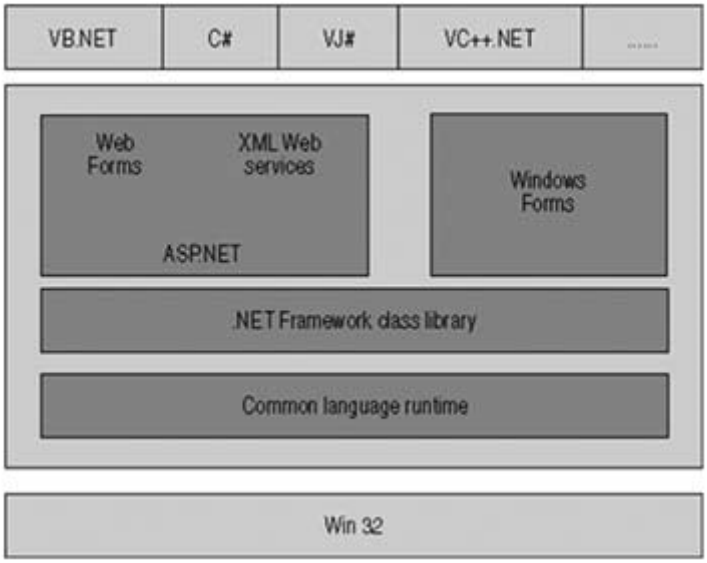
- CLR(Common Language Runtime) یا همان Runtime .

Runtime ، مسئولیت رسیدگی به سرویس ها را برعهده داشته و شامل یکپارچگی زبانها ، امنیت و مدیریت حافظه است . در زمان پیاده سازی ، Runtime ویژگی های لازم و مورد نیازرا با هدف تسهیل در پیاده سازی، ارائه می نماید .

- کتابخانه های کلاس (Libraries – Class)

کتابخانه های کلاس ، امکان استفاده مجدد از کدها برای اکثر عملیات متداول نظیر دستیابی به داده ها ، پیاده سازی سرویس های وب XML ، فرم های وب و یا ویندوز را فراهم می نماید .

معماری دات‌نت:



دات‌نت امکانات متنوعی را برای برنامه‌نویسی در اختیار ما قرار می‌دهد تا برای کامپیوترهای شخصی و موبایل کامپیوترها سیستم‌های قدرتمندی بنویسیم .از مزایای دات‌نت این است که برنامه‌ها و دستگاه‌ها را با استفاده از استانداردهایی همچون XML، HTTP، SOAPیکپارچه می‌سازد.

همچنین دات‌نت زیربنای کنترلی از راه‌دور را برای ما فراهم می‌سازد که به ما اجازه می‌دهد برنامه‌ها را در پروسه‌های مختلف و روی کامپیوترهای مجزا اجرا کنیم و اطلاعات را با استفاده از باینری و پروتکل HTTPجابه‌جا سازیم.

ابزارهایی که دات نت در اختیار ما قرار می‌دهد، عبارت است از :

-وب‌سرویس‌ها

-NET Server Infrastructure

-نرم‌افزارهای Smart Client

-Visual Studio .NET

.NET Frameworkدر حقیقت ساختار زیربنایی برای ساختن برنامه‌های دات‌نت است.

مدل برنامه‌نویسی .NET Frameworkمدلی شیءگرا است. برای ساخت برنامه‌ای در دات‌نت ابتدا کلاس اصلی آن را می‌سازیم .

سپس عملکرد آن را در قالب متدهای آن کلاس قرار می‌دهیم. نکته مهمی که در اینجا باید به آن اشاره نمود این است که کلاسی که مثلاً با کدهای C#.NETنوشته شده است، می‌تواند به راحتی به VB.NETتغییر یابد

ECMAیا European Computer Manufactures Association، زبان عمومی، شامل نقش‌هایی برای این تبدیل شدن کدها و قابلیت Interoperability، را به نام CLRیا Common Language Runtimeتعیین کرده است. از آن جایی که کدهایی که با CLRهماهنگ هستند، موقع کامپایل به کدهایی به نام ILیا Intermediate Languageتبدیل می‌شوند (و این کدها هستند که در واقع اجرا می‌شوند). کدهای برنامه‌هایی که با CLRسازگار هستند، می‌توانند به کدهای دیگر زبان‌هایی که با CLRهماهنگ هستند تبدیل می‌شود.

زبان‌هایی که با CLRهماهنگ هستند، عبارتند از :

- Microsoft Visual Basic.NET
- #Microsoft Visual C
- Microsoft Visual C++.NET
- Microsoft Visual J# .NET

شایان ذکر است که در بین این زبان‌ها C# به‌عنوان زبان استاندارد بین‌المللی توسط ECMAمعرفی شده است .

کدهایی که با این زبان‌ها نوشته می‌شوند، همه به کدهای میانی به نام MSILیا Microsoft Intermediate Language تبدیل می‌شوند .

پس می‌توان با هر یک از زبان های بالا برنامه را آماده کرد و این برنامه می‌تواند با برنامه‌های دیگری که به زبان متفاوت دیگری نوشته شده است، سازگار باشد .

همان‌طور که قبلاً بیان شد یکی از امکاناتی که معماری دات‌نت در اختیار ما قرار می‌دهد به نام Side by Side Execution معروف است. بدین‌معنا که با استفاده از دات‌نت فریم‌ورک برنامه‌نویسان می‌توانند با استفاده از اسمبلی‌ها نسخه‌های چندتایی از یک برنامه را در یک سیستم داشته باشند.

اسمبلی‌ها شامل کدهای ILو) Metadataمسئول نگهداری اطلاعاتی همچون نام و نسخه برنامه) هستند .

CLRدر حقیقت با استفاده از Metadataو جمع‌آوری اطلاعات نسخه‌های برنامه می‌تواند نیازهای برنامه را پیدا کند و به ما اجازه دهد نسخه‌های مختلف از یک برنامه را به صورت Side By Sideاجرا کنیم.

کتابخانه کلاس‌ها و CLRکتابخانه کلاس‌ها در حقیقت Typeهایی که در دات‌نت عمومی هستند را مشخص می‌کنند و با استفاده از این کلاس‌ها برنامه‌نویسان می‌توانند برنامه‌های گوناگون از قبیل Windows Forms، Web Formsو XML

Web Serviceها را تهیه کنند .

از طرف دیگر، قسمت CLRشامل کامپوننت‌هایی است که کدهای ILرا اجرا می‌کند و امنیت، صحت نوع داده‌ها و... را کنترل می‌کند. کدهایی که درون CLRاجرا می‌شوند را اصطلاحاً Managed Codeو اگر خارج از CLRاجرا شوند، Unmanaged Codeمی‌نامند.

زبان مشترک در دات‌نت

همان‌طور که اشاره شد CLRیکی از اجزای مهم فریم‌ورک‌دات‌نت است. کار اصلی CLRایجاد محیط اجرایی برنامه و ارایه سرویس به برنامه‌ها است. در واقع اجزای CLRدائماً با همدیگر در ارتباط هستند و کار می‌کنند تا بتوانند محیطی زیربنایی در دات‌نت برای اجرا شدن مهیا سازند. کار اصلی این کامپوننت‌ها لودکردن کدهای ILبرنامه، کامپایل کردن و اجرای آن کدها است .

کامپوننت‌های CLR عبارتند از :

-لودکننده کلاس‌ها یا Class Loaderبرای Loadکردن کلاس‌ها در حین اجرای برنامه

-کامپایلر تبدیل‌کننده کدهای MSILبه Native Code

-مدیر کدها یا Code Managerکه مسئول مدیریت کدها در زمان اجرای برنامه است .

GB -یا Garbage Collectorبرای مدیریت حافظه

-موتور مدیریت امنیت در سیستم

-چک‌کننده نوع پارامترها

-پشتیبانی‌کننده Threadها

-مدیریتExceptionها

-موتور اشکال‌زدایی (debug) سیستم

COM Marshaler -برای مدیریت تبادل اطلاعات بین برنامه‌های دات‌نت و برنامه‌های COM

Base Class Library-

برای این‌که یک برنامه بتواند در CLRاجرا شود، باید به یکی از زبان‌های .NETو CLS-Compliantنوشته شود .

کامپایلر دات‌نت این کدها را به کدهای MSILو Metadataتبدیل می‌کند .

این کدها شامل اطلاعاتی از قبیل نحوه لود شدن، ذخیره‌شدن و چگونگی فراخوانی متدها در کلاس‌ها می‌شود .

کدهای MSILهمچنین شامل دستورات خاصی برای کار با عملیات ریاضی، چگونگی مدیریت خطاها و... نیز هست.

مدیریت خودکار حافظه در دات‌نت

یکی از مزایای دات‌نت مدیریت خودکار حافظه است که موقع اجرای برنامه‌ها استفاده می‌شود. در دات‌نت رویه‌های اجرایی مدیریت می‌شوند و به اصطلاح موقع اجرای برنامه‌ها MEPیا Managed Execution Processاستفاده می‌شود، ولی به راستی MEPچیست؟

MEPدر حقیقت پروسه‌هایی هستند که در لود شدن و اجرای برنامه‌ها حافظه را به صورت خودکار مدیریت می‌کنند. همچنین سرویس‌های دیگری مانند چک کردن Type Safety، امنیت و مدیریت خطاها را ارایه می‌کنند MEP .در واقع شامل کدهای مدیریت شده (Managed Code) و اطلاعات مدیریت شده (Managed Data) است.

در دات‌نت این اطلاعات در کدهای MSILذخیره می‌شوند و به صورت metadataدر فایل‌های اجرایی exeوجود دارند . Managed Dataتوسط GCیا Garbage Collectorبه حافظه منتقل می‌گردند و موقعی که نیازی نیست از حافظه برداشته می‌شوند .
اگرچه Managed Codeمی‌تواند به Managed Dataو Unmanaged Dataدسترسی داشته باشند، Managed Dataفقط از طریق Managed Codeقابل دسترسی است.

می دانید که زبان های برنامه نویسی مختلفی برای برنامه نویسی در این محیط وجود دارد. از زبان های مایکروسافت مثل سی شارپ و ویژوال بیسیک گرفته تا زبان هایی همچون **Ruby.NET** و **Python for .NET** برنامه نویسان را قادر می سازد تا به زبان مورد علاقه ی خود برای این پلت فرم برنامه بنویسند.

در میان تعداد زیادی زبان برنامه نویسی که برای پلت فرم دات نت وجود دارد، به تازگی زبانی کاملاً فارسی به نام **Farsi.NET** طراحی و پیاده سازی شده است که امکان ایجاد برنامه های شیء گرای تحت دات نت را به زبان کاملاً فارسی به برنامه نویسان می دهد. دستورات فارسی دات نت کاملاً باید به زبان فارسی و از راست به چپ تایپ شوند و قواعد آن بسیار شبیه زبان سی شارپ است. در واقع فارسی دات نت طوری پیاده سازی شده است که ابتدا به زبان سی شارپ ترجمه می شود و سپس با استفاده از کامپایلر سی شارپ به فایل اجرایی تبدیل می گردد. در عکس زیر نمونه برنامه ی ساده ای را به زبان **Farsi.NET** مشاهده می کنید.

ویژوال استادیو محیط توسعه مایکروسافت برای نرم افزارهای مبتنی بر فریم ورک دات نت است. این نرم افزار قدرتمند دارای قابلیت های فراوانی است که برنامه نویسی و توسعه نرم افزاری های تحت ویندوز و تحت وب را آسان تر و لذت بخش تر از همیشه کرده است. برنامه نویسان معمولاً مانند کاربران معمولی از برنامه های کامپیوتری استفاده نمی کنند، آن ها همیشه دنبال راهی برای آسان تر کردن کارها و صرفه جویی در زمان هستند. از آنجا که کاربران ویژوال استادیو برنامه نویسان هستند و اصولاً برنامه نویسان حال و حوصله پیدا کردن یک دستور میان منو های تو در تو و یا شکلک های موجود در نوار ابزارها را ندارند، ویژوال استادیو صادر کردن دستورات را بوسیله کلیدهای میانبر ساده تر نموده است.

همیشه برنامه نویسان جاوا خاصیت مستقل از پلت فرم بودن آن را یکی از مهمترین مزیت های جاوا بر دات نت می دانستند. حق هم داشتند، آن ها یکبار برنامه شان را می نوشتند و به کمک ماشین مجازی جاوا روی هر سیستم عاملی اجرایش می کردند. جاوا با شعار یک زبان، پلت فرم های مختلف به بازار اومد اما دات نت با شعار یک پلت فرم، زبان های مختلف قدم در این رقابت گذاشت. پلت فرم دات نت با گذشت ۴ نسخه از آن، پلت فرم جدیدی محسوب نمی شود و مهمترین و جدی ترین رقیب جاوایی است که الان در دست شرکت اوراکل است. چندسال پیش زمزمه های پروژه ای به گوش رسید که مژده اجرای نرم افزارهای مبتنی بر دات نت را بر روی سیستم عامل های دیگر میداد. **پروژه مونو** یک فریم ورک دات نت cross-platform و اوپن سورس است که به شما اجازه می دهد که برنامه های مبتنی بر دات نت خود را که تا به حال فقط روی ویندوز اجرا می شده را روی سیستم عامل های مبتنی بر لینوکس و حتی سیستم عامل شرکت اپل اجرا کنید. آخرین نسخه پایدار مونو از C# 3.0، VB 8، ASP.NET 2.0, ADO.NET و Windows Forms 2.0 پشتیبانی می کند. در واقع اگر برنامه خودتان را با دات نت فریم ورک نسخه ۲.۰ نوشته باشید و از APIهای ویندوز نیز استفاده نکرده باشید، به راحتی می توانید آن را روی لینوکس یا مک اجرا کنید

. برنامه نویسان دات نت نیز می توانند از Python برای کدنویسی استفاده کنند. یکی از ویژگی های بسیار خوب دات نت پشتیبانی از زبان های مختلف برنامه نویسی است که از این نظر برتری نسبی در مقابل رقیبان خود دارد. در سال ۲۰۰۵ زبان برنامه نویسی **IronPython** برای آوردن زبان محبوب Python به دنیای دات نت ساخته شد و حالا استفاده از دات نت فریم ورک و سیلورلایت را برای برنامه نویسان Python امکان پذیر کرده است. نسخه جدید IronPython بر پایه DLR بنا شده است. DLR افزونه ای بر روی CLR است که پیاده سازی زبان های داینامیک را آسان می کند. قواعد برنامه نویس پایتون بسیار آسان است و برنامه های نوشته شده به این زبان بسیار واضح هستند.

مقایسه دات نت و جاوا :

J2EE یک استاندارد برای تحقق یکApplication Server " است در حالیکه دات نت یک " نرم افزار "است که" فقط کلاسهای پایه برای تولید نرم افزارهای مستقل یا مرتبط شبکه ای را در اختیار توسعه گر قرار می دهد.

دات نت : یعنی یک بستر برای ایجاد نرم افزار . یک بار توسط مایکروسافت نوشته شده و بقیه باید ازآن استفاده کنند اما تمام دات نت چیزی بیشتر از یک "بستر اجرای کد و کتابخانه مقدماتی کلاس" نیست.تمام دات نت یک FrameWork است و یک محیط تولید نرم افزار + مستنداتی جهت ایجاد ابزارهای متفرقه تولید نرم افزار.

J2EE: یک استاندارد است که مشخص میکند برای پاسخ دادن به یک نیاز نرم افزاری "سازمان مقیاس "چگونه باید با اجزاء نرم افزار رفتار کرد و برای مدیریت طول عمر نرم افزار چه کار باید کرد . این استاندارد توسط سان ارائه شده است. خیلی ها مبتنی بر این استاندارد نرم افزارهای خودکار سازی ایجاد کرده اند. استاندارد J2EE میگوید چگونه با" زبان جاو ا "یک FrameWork ایجاد کنیم چگونه کتابخانهء کلاس برای تمام مقاصد بنویسیم بانک اطلاعاتی چطور باشد و.. و شر کتهای متعددی بر اساس این استاندارد Application Server های مبتنی بر J2EE ایجاد کرده اند که برخی تمام J2EE را پیاده سازی کرده اند و برخی فقط بخشی آن را.

ApplicationServer یک بستهء نرم افزاری است که وظیفه اش Application LifecycleManagement است . یعنی از ابتدای تولد یک نرم افزار "سازمان مقیاس" تاانتهای اتمام تولید ، باید به تمام نیازهای نرم افزاری پاسخ دهد . یعنی اگر برنامه نویس به یک FrameWorkاحتیاج داشت ، Application Serverیک Frameworkبه او بدهد ، اگر بانک اطلاعاتی خواست ، Application Serverیک بانک اطلاعاتی کامل برایش فراهم کند ، اگر وب سرور خواست ، ApplicationServerیک وب سرور تمام عیار به او بدهد ، اگر برای ارسال نامه های الکترونیکی برنامه اش به یک SMTP سرور نیاز داشت ، Application Serverیک سرور SMTP به او بدهد ، اگر قرار شد برای احراز هویت از Kerberos استفادهکند یک پیاده سازی کربرایزد از استک TCP/IP در Application serverوجودداشته باشد ، اگر خواست برای منطق محاسباتی برنامه اش یک GUI ایجاد کند ، Application Serverیک IDE و GUI Builderبه او بدهد ، اگر خواست داده های کاربری را از کارتهای هوشمند (smart Card) دریافت کند ، رابطهای لازم و API های مربوطه را از Application Server بگیرد و ... به دیگر بیان Application Serverیک محیط Integrated است برای طراحی و تولید و مدیریت وتوزیع و کاربرد یک نرم افزار "سازمان مقیاس" . شاید این سوال در ذهن عده ای ایجاد بشود که مگر تمام اجزاء یک Application Serverرو نمیشود بصورت منفرد پیدا کرد ؟ وب سرور ، GUI Builder، سرور پست الکترونیکی ، توابع احراز هویت ، بستر اجرای کد و ... ؟ جواب مثبت است اما چه کسی میتونه تضمین کند تمام این اجزاء دارای "سطح" ی یکنواخت و یکسان باشند ؟ (InnovativeIntegrated Interfaceیا مثلا" چه کسی می تواند تضمین کنه یک بستر اجرای کدبتواند توابع دسترسی به بانک اطلاعاتی رو با بهینه ترین وضعیت تولید کند ؟یا تضمین کند این عناصر با هم سازگاری مناسبی داشته باشند ؟ همگی تولید شده توسط یک بستر خاص باشند که توسط همان بستر بشود بین آنها ارتباط برقرارکرد ؟ اینجا مسئله سازگاری است . یعنی اگر قرار شد یک گروه نرم افزاری برای بزرگترین سازمان بیمه غیر دولتی امریکا یک راهکار جامع ERP تولید کند (یابخرد و خصوصی سازی کند برای محیطش) باید به چه بستری اعتماد کند که مطمئن باشد تمام درخواستهای نرم افزاری " سازمان مقیاس" ش را می تواند جواب بده ومشخصه های اون ، سازمانش رو به یک نرم افزار خاص ، سخت افزار خاص ، پروتکل خاص و ... محدود نمیکند ؟ (اصولا" محدودیت در ادبیات آی تی

، سطح اعتمادو قابلیت وثوق - Reliability - رو کاهش میدهد) اینجاست که یک ApplicationServer خودنمائی میکند . یک Application Serverتضمین میکند که از بستراجرای کد گرفته تا وب سرور ، از توابع امنیتی گرفته تا بانک اطلاعاتی ، از IDEگرفته تا ابزارهای حمایت از) UP یا (Unified Process و ... در بستهءنرم افزاریش وجود دارد.

J2EEدر واقع یوتوپیا (آرمان شهر) شرکت سان مایکروسیستمز است برای تولید یکApplication Server

شرکت سان غیر از ارائه یوتوپیای یک Application Serverکارهای دیگری هم انجام داده . مثلا" توسعه زبان جاوا . طبیعی است که زبان جاوا زبان استانداردتوسعه نرم افزارهای مبتنی بر J2EE است ، هر چند بر خلاف اظهارات ناشیانه برخی ، J2EEو خصوصا" بستر اجرای کدش ، به زبان جاوا منحصر نیست . یعنی همانطور که [مثلا"] بستر دات نت قابلیت پذیرش زبانهای مختلف رو دارد ،بسترهای مبتنی بر جاوا هم می توانند به سایر زبانهای برنامه نویسی سرویس بدهند. یعنی براحتی می شود بین جاوا و سایر کتابخانه هائی که توسط سایر زبانهای برنامه نویسی تولید شده ارتباط برقرار کرد (Java Native Interface) . هرچند که مثل دات نت منعطف نیست . سان تلاش میکند یک ApplicationServer مبتنی بر استاندارد خودش یعنی J2EE تولید کنه اما هنوز تکمیل نشده است. (سان فعالیت گسترده ای برای توسعه خود جاوا و بهینه سازی منطق J2EE وکلاسهای تولید نرم افزار دارد ، فعالیتهای خفنی در هم در عرصه سخت افزاردارد)اما نگارشهای عملیاتی متعددی از Application Server های مبتنی بر J2EE وجود دارد که فقط یکی از انهاتمام جزئیات رو پیاده سازی کرده است.

الف. کتابخانه کلاس : داتنت و JDK هر دو کتابخانه های قدرتمندی هستند که اغلب نیازهای پایه برای تولید نرم افزار را حمایت میکنند . اما برای تولید نرم افزارهای بزرگ مقیاس هیچکدام کافی نیستند . دات نت چیز دیگری ندارد اما برای جاواراهکارهای دیگری هم وجود دارد . کاربری دات نت راحت تر است . پیچیدگی های دات نت هم کمتر است . نمودار یادگیری جاوا بسیار کم شیب است.

ب. زمان اجرا: زمان اجرای دات نت تقلیدی صرف از زمان اجرای جاوا ست در هر دو محیط خوب است . سرعت اجرای "برنامه" های دات نت از برنامه های جاوا کندتر است اگر از JIT استفاده نکند. سیستم Code Caching و JITC دات نت کمک زیادی به افزایش سرعت برنامه ها کرده است .

ج. اتصالات : داتنت از ریموتینگ ، وب سرویس و کام پلاس حمایت میکند (بصورت داخلی) .جاوابجای ریموتینگ چیزی بنام ریموت متد اینووکیشن دارد ، وب سرویس رو حمایت میکند ، CORBAرو حمایت میکند ، چیزی بنام EJB دارد که اشیاء شناور در یک"مخزن سازمانی" هستند که افراد ، سرویسها و نرم افزارها بنا به میزان دسترسی می توانند از آن استفاده کنند . کنترلهای دات نت هنوز چنین قابلیتی ندارند و دات نت هنوز راهی برای ایجاد یک Object Repository سازمانی ارائه نکرده . اشیاء کام پلاس و محیط MTS ویندوز هم (با اینکه ربطی به دات نت ندارند بطور مستقیم) مانند EJB ها منعطف نیستند EJB . ها State-Lessنیستند.

د. ارتباط با داده : داتنت چیزی بنام ADO .NET ارائه کرده که راه حلی است منحصر به فرد . جاوا JDBCرو داره که چه در connection Poolingو چه در objecy pooling به خوبی ADO .NETکار میکنه اما ADO .NET فوق العاده امکانات زیادی دارد

نتیجه گیری :

خیلی ها فکر می کنند هر چه یادگیری یک زبان برنامه نویسی سخت تر باشد، حتماً بر زبان های دیگر برتری دارد. به طور مثال یکی از سخت ترین زبان های برنامه نویسی سی++ است. به نظر شما یک برنامه نویس سی++ چه برتری بر یک برنامه نویس ویژوال بیسیک دات نت دارد!؟ تفاوت ها نسبی است. مثلاً برنامه نویس سی++ می تواند یک درایور سخت افزاری بنویسد اما برنامه نویس وی بی دات نت نمی تواند. حالا به نظر شما کدام بهتر می توانند یک وب اپلیکیشن بنویسند!؟ کدام سریعتر می تواند یک نرم افزار تولید کند!؟ کدامیک ابزارهای قوی تری برای توسعه در اختیار دارد!؟ خروجی کدامیک سریعتر اجرا می شود!؟

اخیراً بسیاری از برنامه نوسان سی++ به جاوا و سی شارپ روی آورده اند. چرا!؟ مگر سی++ سخت تر نیست! پس چرا این ها به یک زبان ساده تر روی آورده اند!؟

پاسخ این است که در دنیای امروز، **برنامه نویسان باید پلت فرم محور باشند نه زبان محور**. زبان برنامه نویسی هر چقدر هم که قوی و سخت باشد، اگر توسط یک پلت فرم قدرتمند و شناخته شده پشتیبانی نشود هیچ ارزشی نخواهد داشت. به طور مثال در سیستم عامل لینوکس از سی++ به خوبی پشتیبانی می شود اما در وب اینطور نیست. این سی++ را به یک انتخاب مناسب برای نوشتن برنامه های مبتنی بر لینوکس و به یک انتخاب نا مناسب برای نوشتن برنامه های مبتنی بر وب تبدیل می کند. PHP در وب به خوبی پشتیبانی می شود و برای توسعه برنامه های تحت وب بسیار مناسب است اما در بقیه موارد حرفی برای گفتن ندارد.

البته کسی که سی++ بلد باشد و بعد از آن مثلاً PHP را شروع کند خیلی سریعتر و بهتر از کسی که قبلاً مثلاً با ویژوال بیسیک آشنا بوده،PHP را یاد می گیرد. همانطور که ذکر شد پلت فرم از زبان خیلی مهم تر است. شما می توانید در عرض ۳ ماه سی++ را به طور کامل یاد بگیرید، اما نمی توانید در فقط ۳ ماه یک برنامه نویس وب حرفه ای شوید. یادگیری سینتاکس یک زبان برنامه نویسی هر چقدر هم که مشکل باشد، در ۳ یا ۴ ماه میسر می شود اما یادگیری یک پلت فرم برنامه نویسی مانند دات نت و حرفه ای شدن در آن به این سادگی ها نیست!

کسانی که خود را محدود به یک زبان برنامه نویسی می کنند و روی آن تعصب دارند هیچ وقت موفق نخواهند شد. کسانی هستند که برای تولید نرم افزار از جاوا و اوراکل استفاده می کنند اما نرم افزارشان به اندازه نرم افزاری که با دلفی و اینتربیس تولیدشده، بازدهی مطلوب ندارد.

در دنیای امروز باید بر اساس نیاز و خواست مشتری از این ابزارها استفاده کرد.