بررسی J2EE و .Net

قبل از اینکه شروع کنم چند تا نکته مقدماتی لازمه ذکر بشه تا ابهام موجود در ذهن بعضی از دوستان بر طرف بشه منطقا" مقایسه دات نت و J2EE یه چیزی است تو مایه های مقایسهء نوت بوک و MainFrame . (عمدا" از این دو لفظ برای تشبیه استفاده کردم . بعدا" مشخص میشه چرا) هر دو به نوعی کامپیوتر هستند ، یعنی محاسبه کننده و هر دو از سخت افزارهائی تحت کنترل نرم افزار تشکیل شده اند اما هر کدام برای هدف خاصی طراحی شده اند و برای کاربرد خاصی مناسب هستند . مقایسه های مختلفی بین این دو تکنولوژی روی اینترنت موجوده که من به دو دسته تقسیمشون میکنم . گروه اول کسانی هستند که طرفدار یکی از دو تکنولوژی هستند (به هر دلیلی) و قصد دارند با براهین و ادله بعضا علمی و بعضا غیر علمی برتری یکی رو بر دیگری اثبات کنند و گروه دوم کسانی هستند که بدون تعصب اما با نیت مشخص کردن برتریهای هر تکنولوژی بر دیگری در حوزه ای خاص تولید شده اند مثلا" یکی میگه ASP.NET برای تولید محتوای وب بهتره و RMI برای ایجاد ارتباط Multi Tire و ... و فی الواقع بگن هر دو چیزهای خوبی دارند اما هر کدام در یک بخش . "نظر شخصی" من اینه که هر دو گروه اشتباه میکنند . اول در مورد علت این طرز فکر توضیح میدم ، بعد در مورد بخشهای مختلف هر دو تکنولوژی یه سری مطلب عرض میکنم .

سوال : چرا فکر میکنم میشه به دات نت و J2EE به عنوان دو موجودیت مستقل نگاه کرد و هر دو رو بررسی کرد اما مقایسه نکرد ؟

جواب : بخاطر مؤلفه های ذاتی هر کدام از این تکنولوژی ها . J2EE یک استاندارد برای تحقق یک Application Server است در حالیکه دات نت یک " نرم افزار" است که "فقط" کلاسهای "پایه" برای تولید نرم افزارهای مستقل یا مرتبط شبکه ای رو در اختیار توسعه گر قرار میده .

دات نت : یعنی یک بستر برای ایجاد نرم افزار . یک بار توسط مایکروسافت نوشته شده و بقیه باید از اون 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 (e.g : compiler)

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 (Application

Lifecycle Management) چکار باید کرد . این استاندارد توسط سان ارائه شده . خیلی ها مبتنی بر این استاندارد نرم افزارهای خودکار سازی ایجاد کرده اند ، سان مایکروسیستمز نیز هم . استاندارد J2EE میگه چگونه با "زبان جاوا" یک FrameWork ایجاد کنیم ، چگونه کتابخانهء کلاس برای تمام مقاصد بنویسیم ، بانک اطلاعاتی چطور باشه و ... و شرکتهای متعددی بر اساس این استاندارد Application Server های مبتنی بر J2EE ایجاد کرده اند که برخی شون تمام J2EE رو پیاده سازی کرده اند و برخی شون فقط بخشی از اون رو . در مورد جزئیات Application Server های J2EE کمی بعد توضیح میدم .

سوال : پس احتمالا" کلید درک تفاوت ماهیتی دات نت و J2EE در درک صحیح از مفهوم Application Server خلاصه میشه ، درسته ؟

جواب : بله . Application Server یک بستهء نرم افزاری است که وظیفه اش Application Lifecycle Management است . یعنی از ابتدای تولد یک نرم افزار "سازمان مقیاس" تا انتهای اتمام تولید ، باید به تمام نیازهای نرم افزاری پاسخ دهد . یعنی اگر برنامه نویس به یک FrameWork احتیاج داشت ، Application Server یک Framework به او بدهد ، اگر بانک اطلاعاتی خواست ، Application Server یک بانک اطلاعاتی کامل برایش فراهم کند ، اگر وب سرور خواست ، Application Server یک وب سرور تمام عیار به او بدهد ، اگر برای ارسال نامه های الکترونیکی برنامه اش به یک SMTP سرور نیاز داشت ، Application Server یک سرور SMTP به او بدهد ، اگر قرار شد برای احراز هویت از Kerberos استفاده کند یک پیاده سازی کربرایزد از استک TCP/IP در Application server وجود داشته باشد ، اگر خواست برای منطق محاسباتی برنامه اش یک GUI ایجاد کند ، Application Server یک IDE و GUI Builder به او بدهد ، اگر خواست داده های کاربری رو از کارتهای هوشمند (smart Card) دریافت کند ، رابطهای لازم و API های مربوطه را از Application Server بگیرد و ... به دیگر بیان Application Server یک محیط Integrated است برای طراحی و تولید و مدیریت و توزیع و کاربرد یک نرم افزار "سازمان مقیاس" . شاید این سوال در ذهن عده ای ایجاد بشه که مگه تمام اجزاء یک Application Server رو نمیشه بصورت منفرد پیدا کرد ؟ وب سرور ، GUI Builder ، سرور پست الکترونیکی ، توابع احراز هویت ، بستر اجرای کد و ... ؟ جواب مثبته اما چه کسی میتونه تضمین کنه تمام این اجزاء دارای "سطح" ی یکنواخت و یکسان باشند ؟ (Innovative Integrated Interface) یا مثلا" کی میتونه تضمین کنه یک بستر اجرای کد بتونه توابع دسترسی به بانک اطلاعاتی رو با بهینه ترین وضعیت تولید کنه ؟ یا تضمین کنه این عناصر با هم سازگاری مناسبی داشته

باشن ؟ همگی تولید شده توسط یک بستر خاص باشند که توسط همون بستر بشه بین اونها ارتباط برقرار کرد ؟ اینجا مسئله سازگاریه . یعنی اگر قرار شد یک گروه نرم افزاری برای بزرگترین سازمان بیمه غیر دولتی امریکا یک راهکار جامع ERP تولید کند (یا بخرد و خصوصی سازی کند برای محیطش) باید به چه بستری اعتماد کنه که مطمئن باشه تمام درخواستهای نرم افزاری " سازمان مقیاس" ش رو میتونه جواب بده و مشخصه های اون ، سازمانش رو به یک نرم افزار خاص ، سخت افزار خاص ، پروتکل خاص و ... محدود نمیکنه ؟ (اصولا" محدودیت در ادبیات آی تی ، سطح اعتماد و قابلیت وثوق – Reliability – رو کاهش میده) اینجاست که یک Application Server خودنمائی میکنه . یک Application Server تضمین میکنه که از بستر اجرای کد گرفته تا وب سرور ، از توابع امنیتی گرفته تا بانک اطلاعاتی ، از IDE گرفته تا ابزارهای حمایت از UP (یا Unified Process) و ... در بستهء نرم افزاریش وجود داره .

سوال : خوب حالا با این توصیفات J2EE دقیقا" چیه ؟

جواب : J2EE در واقع یوتوپیا (آرمان شهر) شرکت سان مایکروسیستمز است برای تولید یک Application Server.

نتیجه : فکر میکنم مطالب بالا باعث شده باشه این تصور غلط که میشه بین دات نت و J2EE مقایسه ای وجود داشته باشه ، از بین برده باشه .

ابسترکت : چه زمانی J2EE مناسب است ؟

شرکت سان غیر از ارائه یوتوپیای یک Application Server کارهای دیگری هم انجام داده . مثلا" توسعه زبان جاوا . طبیعیه که زبان جاوا زبان استاندارد توسعه نرم افزارهای مبتنی بر J2EE باشه ، هر چند بر خلاف اظهارات ناشیانه برخی ، J2EE و خصوصا" بستر اجرای کدش ، به زبان جاوا منحصر نیست . یعنی همونطور که [مثلا"] بستر دات نت قابلیت پذیرش زبانهای مختلف رو داره ، بسترهای مبتنی بر جاوا هم میتونن به سایر زبانهای برنامه نویسی سرویس بده . یعنی براحتی میشه بین جاوا و سایر کتابخانه هائی که توسط سایر زبانهای برنامه نویسی تولید شده ارتباط برقرار کرد . (Java Native Interface) هر چند که مثل دات نت منعطف نیست . سان داره تلاش میکنه یک Application Server مبتنی بر استاندارد خودش یعنی J2EE تولید کنه اما هنوز تکمیل نشده . (سان فعالیت گسترده ای برای توسعه خود جاوا و بهینه سازی منطق J2EE و کلاسهای تولید نرم افزار داره ، فعالیتهای خفنی در هم در عرصه سخت افزار داره ، نباید انتظاری بیش از این ازش داشت) اما

نگارشهای عملیاتی متعددی از Application Server های مبتنی بر J2EE وجود داره که فقط یکی از اونها تمام جزئیات رو پیاده سازی کرده .

معرفی Application Server اوراکل به عنوان جامعترین Application Server :

در ایران اغلب اوراکل رو به عنوان یک بانک اطلاعاتی میشناسن در حالیکه بانک اطلاعاتی اوراکل فقط بخشی از اون چیزی است که اوراکل تحت عنوان e-Bussines Suite منتشر کرده . Application Server اوراکل تمام اون چیزهائی که در وصف یک Application Server عرض کردم داره . بطور مختصر و لیست وار در موردش توضیحاتی عرض میکنم تا کمی روشنتر بشه بحث :

۱. یک بانک اطلاعاتی کامل : اوراکل فعلا" تنها بانک اطلاعاتی است که نه تنها نسخه های متعددی برای MainFrame ها داره ، برای تمام بسترهای نرم افزاری و سخت افزاری موجود هم نسخه هائی رو ارائه کرده . بزرگترین بانک اطلاعاتی که این حقیر در جریانش هستم و با اوراکل کار میکنه بانک اطلاعاتی وزارت انرژی ایالات متحده آمریکاست که روی یک MainFrame شرکت IBM اجرا شده . بانک اوراکل یک نسخهء کامل زبان پرس و جوی ساخت یافته یعنی PL/SQL ، یک سوئیت کامل بنام PSP که برای تولید صفحات وب بطور مستقیم از PL/SQL استفاده میکنه است . اوراکل تنها بانک اطلاعاتی است که موتور آن (DB Engine) هم میتونه بصورت توزیع شده و چند بخشی (Clustered) اجرا بشه . حتی میشه بخضی از انجین رو روی یک بستر کوچک وینتل(ویندوز + اینتل) و بخشی دیگر رو روی یک ماشین غول پیکر HP مجهز به HP-UX اجرا کرد . حتی میشه حین سرویس دهی بانک ، بانک رو از یک پلت فرم به پلت فرم دیگه منتقل کرد . (ویژگی های منحصر به فردش رو عرض کردم)

۲. یک بستر اجرای کد نرم افزار : Application Server شرکت اوراکل بطور کامل "بخش نرم افزاری J2EE " یعنی کتابخانه های کلاسش رو پیاده سازی کرده .

JDBC Connectors
JSP Engine
JavaBeans Engine
RMI
JMS
JINI
JMX
JIRO

J2EE CORBA ORB
JXTA
JXML
JCP
JNI
Web Service Implementation
و ...

یعنی هر کسی هر برنامه ای مبتنی بر J2EE نوشته باشه در بستر Application Server اوراکل قابل اجرا و سرویس دهی است . اوراکل J2SE و J2ME رو هم حمایت میکنه (دومی برای تولید برنامه های موبایل برای پورتابل دیوایسها کاربرد داره) . همچنین اوراکل بطور کامل یک نسخه از Java Smart Card API رو پیاد سازی کرده . در حال حاضر جاوا تنها ابزاری است که میشه توسط اون تقریبا" برای تمام کارتهای هوشمند برنامه نوشت ضمن اینکه توسط قابلیت فوق الذکر قطعه کدهای قابل ذخیره سازی در کارتهای هوشمند هم قابل تولید است . فرض کنید یک تابع تبدیل تاریخ مینویسید و تابع رو داخل کارت هوشمند قرار میدید ، هر وقت نرم افزار اون تابع رو صدا زد کارت رو در کارت خوان میگذارید و برنامه شما تابع رو روی کارت هوشمند صدا میزنه و جواب میگیره بدون اینکه در مورد پیاده سازی اش چیزی بدونه .

۳. اوراکل یک وب سرور مخصوص به خود ، همچنین سرورهای :

POP3
SMTP
FTP
WebDav
Cache Server
Common Internet File system - CIFS
LDAP compatible Directory Service
و...

رو بطور کامل پیاده سازی کرده . تمام این سرورهای نرم افزاری کاملا" با هم سازگار هستند و برای کار روی یک محیط مبتنی بر J2EE بهینه سازی و خصوصی سازی شدن .

۴. اوراکل یک content Management System داره که قابلیت ایجاد پورتال های مبتنی بر وب روی اینترنت یا اینترانت رو به "نرم افزار" های J2EE میده .

۵. اوراکل یک محیط کامل تولید برنامهء کاربردی بنام اوراکل J Developer داره که یک IDE و GUI Builder کامل است .

۶. اوراکل یک سرویس (یعنی نرم افزارهائی + سرویس دهنده هائی) برای ایجاد ویژوال گزارش از بانک اطلاعاتی داره . گزارشها میتونن طراحی بشن تا از داده ها استفاده کنن و خروجی بدن ، یا یک سرویس تولید گزارش به یک نرم افزار متصل بشه تا در زمان اجرا مولفه های گزارش به سرویس گزارش درخواست داده بشن تا گزارش رو طراحی کنه ، به داده متصل کنه و خروجی بده . بهش میگن Reporting Service

۷. ابزارهای مدیریتی قدرتمند برای کنترل تراکنشهای بانک اطلاعاتی خارج از محیط بانک (منحصر به فرد) ، کنترل وضعیت اشیاء مثلا" EJB ها و سطح دسترسی آنها ، انتقال سرویسهای از یک پلت فرم به پلت فرم دیگر بدون توقف روند سرویس دهی ، صف گذاری منطقی و مدیریت شده درخواستها و ...

۸. Load Balancer اوراکل کمک میکنه سرویسها ، بانکهای اطلاعاتی و سرورها و سایر نرم افزارهای مبتنی بر وب یا شبکه روی یک بستر توزیع شده اجرا بشن و اگر فشار ترافیک روی یک سرور زیاد بود ، Load Balancer درخواستها رو به سایر سرورها که توسط قابلیت Replication Service اوراکل بصورت mirror آماده هستند هدایت میکنه . این Load Balancer قابلیت درک جلسات کاربری (Session) ها یا مثلا" متغیرهای سطح برنامه (Application - Level Variables) رو داره . یعنی اگر شما به کتابخانهء ملی سنای امریکا (Pwered By Sun) لاگ این کنید و در حال انتقال صفحاتی از یک کتاب به دایرکتوری شخصی خودتون باشید و فشار روی سرور بانک اطلاعاتی زیاد بشه ، درخواستهای بعدی شما بصورت خودکار به سرور خلوت تری ارسال میشن بدون اینکه State-Less بدون محیط به کانال ارتباطی شما لطمه بزنه ، یعنی هویت شما و Session شما همچنان معتبر است اما روی یک سرور دیگر (این منحصر به فرد نیست اما فقط شرکت مکرومدیا در JRUN که اون هم یک Application Server نصفه نیمه است چنین چیزی داره که در مورد اون هم مطالبی عرض میکنم)

نتیجه اول : اگر شما یک Application Server کامل و قابل اتکاء میخواهید باید بستهء نرم افزاری فوق العاده گران قیمت Oracle 11i - e bussines Suite رو تهیه کنید که هر انچه ذکر شد داخلش موجوده .

نتیجه دوم : برای پاسخ دادن به نیازهای "خیلی بزرگ" و " سازمان مقیاس" که هزینه های میلیونی و میلیاردی براش در نظر گرفته میشه و خطا در انتخاب معماری یا عدم سازگاری اجزاء نا بخشودنی است فقط باید از یک Application Server استفاده کرد که ضمن دارا بودن همه چیز یکجا ، سازگاری ، قابلیت اتکاء و وثوق ، یک شرکت بزرگ و عریض و طویل ازش حمایت کنه . اوراکل یکی از پیشنهادهای موجوده . در بازار E- bussines دنیا اوراکل قدرتمند ترین فروشنده نرم افزاره ، فقط به عنوان مثال مراجعه کنید به خبری از اوراکل که یک پیشنهاد " نه میلیارد دلاری" برای خرید شرکت People Soft (تولید کنندهء ERP های قدرتمند) داده . لازم به ذکره ۹ میلیارد دلار ، کمی کمتر از نصف بودجهء یکسال جمهوری اسلامی ایران است !!

یاد آوردی اول : یک بستر اجرای کد مثل دات نت ، با یک استاندارد برای تولید Application Server یعنی J2EE قابل قیاس نیست .

یاد آوری دوم : داخل یک Application Server یک بستر اجرای کد هم وجود داره که سرویسهای خاصی رو ارائه میکنه . شاید بشه این بسترهای مختلف موجود رو با هم مقایسه کرد .

< اتمام مبحث Application Server ها >

سوال : اگر انچه دات نت ارائه میکنه و به همون "شکل" در بسترهای مبتنی بر J2EE هم موجوده مقایسه کنیم چه نتیجه ای میگیریم ؟

جواب : من دو جور میتونم به این سوال جواب بدم . یکی مبتنی بر تجربه شخصی . یکی مبتنی بر صرفا" ذکر آیتمهای قابل قیاس در مقابل هم .

جور دوم (!) :

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

اگر نمودار عمودی پیشرفت باشه و نمودار افقی زمان) در حالیکه یادگیری دات نت خیلی سریعتر است .

ب. زمان اجرا : زمان اجرای دات نت تقلیدی صرف از زمان اجرای جاوا ست . هیچ بحثی هم درش نیست . حتی کسانیکه مثل بنده عقلشون کم باشه و بشینن و IL رو با ByteCode مقایسه کنن درک خواهند کرد که مایکروسافت خلاقیتی از خودش نشون نداده . JIT در هر دو محیط خوب است . سرعت اجرای "برنامه" های دات نت از برنامه های جاوا کندتر است اگر از JIT استفاده نکند . این حقیقت رو هر کسی با چند آزمایش کوچولو میتونه درک کنه . سیستم Code Caching و JITC دات نت کمک زیادی به افزایش سرعت برنامه ها کرده . جاوا با عمر طولانی اش به ادعای اسکات مک نلی حدود پنجاه بار بهینه سازی شده در حالیکه دات نت هنوز جوونه . به نظر میاد در این یک مقوله باید منتظر آینده شد . اما فی الحال وضع دات نت در این راستا خوبه .

ج. اتصالات : دات نت از ریموتینگ ، وب سرویس و کام پلاس حمایت میکنه (بصورت داخلی) . جاوا بجای ریموتینگ چیزی بنام ریموت متد اینووکیشن داره ، وب سرویس رو حمایت میکنه ، CORBA رو حمایت میکنه ، چیزی بنام EJB داره که اشیاء شناور در یک "مخزن سازمانی" هستند که افراد ، سرویسها و نرم افزارها بنا به میزان دسترسی میتونن ازش استفاده کنن . 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 نداره بطور مستقیم) مانند EJB ها منعطف نیستند . EJB ها State-Less نیستند .

د. ارتباط با داده : دات نت چیزی بنام ADO .NET ارائه کرده که راه حلی است منحصر به فرد . جاوا JDBC رو داره که چه در connection Pooling و چه در objecy pooling به خوبی ADO .NET کار میکنه اما ADO .NET فوق العاده امکانات زیادی داره . چون اینجا دات نت کار زیاده لزومی به توضیح نیست . من با تمام وجود به ADO .NET اعتقاد دارم و تصور نمیکنم معادلی داشته باشه (یکسال و خورده ای پیش چند مقاله کامل در مورد ADO .NET در سایت ایران دولوپرز نوشتم که این مطلب رو اونجا هم عرض کردم . یکسال قبل)

ه. امنیت : امنیت در این حوزه رو "من" به سه بخش تقسیم میکنم (تقسیم بندی کاملا شخصی و تجربی)

ه.۱) حفاظت از متن کد

ه.۲) حفاظت از ارتباطات

ه.۳) حفاظت از خود بستر و حفظ مانائی

در مورد اول هر دو محیط ضعیف هستند . Obfuscator ها نمیتونن به مفهوم واقعی از کد حفاظت کنند و راهکارهای Third party موجود هم بیشتر به طنز شبیه هستند . با داشتن IL یا بایت کد براحتی کد اصلی یا کدی " با قابلیت های" کد اصلی قابل باز-تولید است . پسوردها ، اعداد خصوصی ، کلمه های عبور و ... براحتی قابل بازیافت هستند اگر در متن نرم افزارهای Managed دات نت یا برنامه ها جاوا بکار رفته باشند . اینجا واقعا هیچ ایمنی " نمیتواند" وجود داشته باشد .

در مورد دوم هر دو محیط با Open Standard ها کار میکنند . از SSL گرفته تا Kerberos و از ارتباط با Directory Service ها گرفته تا CA . در این مورد تفاوتی وجود نداره .

در مورد سوم تا حالا مستندی که بر قوت یا ضعف یکی دلالت کنه نخوندم (نمیگم نیست ، نخوندم) و تجربه شخصی و عملی هم ندارم .

فی المجموع در حوزه امنیت دو محیط چندان متفاوت نیستند .

و. انتقال : جاوا از MainFrame ها تا کارتهای هوشمند رو حمایت میکنه . دات قراره بزودی بسترهای دیگه رو حمایت کنه . پس اصولا" در این زمینه هیچ رقابتی وجود نداره . جاوا پانزده سال جلو تره . من با مونو (که قراره بشه دات نت روی لینوکس) کار کردم و فعلا" ناقص و غیر قابل اعتماده . مایکروسافت هم یقینا" تا انتهای ۲۰۰۵ هیچ نسخه ای از دات نت مبتنی بر NIX* ها توزیع نخواهد کرد .

ز. تولید محتوای وب : دات نت .NET ASP رو ارائه کرده . جاوا JSP رو . سرعت پاسخگوئی دات نت در کاربردهای معمولی بالاتره . اما با توجه به محدودیت ویندوز (به عنوان تنها بستر دت نت) برای حمایت از ترافیک و فشار بالا ، اگر کاربردهای خیلی سنگین مد نظر باشه .NET ASP نمیتونه حرفی داشته باشه . موتور JSP هم قابلیت Clustring داره و میشه مجموعه ای از سرورها رو با " یک موتور" راه اندازی کرد . (میدونم به بحث ربطی نداره اما یکبار یکی ازم پرسید چرا مایکروسافت برای MSN و هات میل از فری بی اس دی استفاده میکنه ؟ و نه ویندوز ؟ جواب بنده این بود دلیل هر

چیزی هست ربطی به امنیت نداره . سایت خود مایکروسافت با ترافیک بالا و دشمنانی قسم خورده بدون مشکل داره روی ویندوز کار میکنه . اما وقتی قرار باشه بخاطر ترافیک خیلی بالای مسنجر و ایمیل ، از یک ماشین با مثلا" ۳۰ تا پردازنده استفاده بشه تجربه ویندوز چندان موفقیت آمیز نیست ! در حالیکه فری بی اس دی - اچ پی یو ایکس و سولاریس همین حالا روی ماشینهائی بیشتر از پنجاه پردازنده هم خوب کار میکنند . سان سرور بنام K Fire۱۵ داره - ر.ک بخش سرورهای سایت سان - به قیمت " ده میلیون دلار" میفروشتش و ۱۰۵ تا پردازنده ۶۴ بیتی داره و همین نسخهء سولاریس معمولی روی اون هم کار میکنه و جواب میده و توانائیش ۶۵۰۰ میپسه ! یعنی ۲۰۰۰ میپس قوی تر از بزرگترین مین فریمه IBM - تاریخ این امار متعلق به یکسال پیشه که من پروژه ای داشتم در این مورد)

نتیجه : برای کاربردهای عمومی وب یعنی اونچیزی که در ۹۹ درصو اوقات مد نظره ASP .NET بهتر است مگر اینکه برنامهء خاصی برای انتقال وجود داشته باشه یا احتمال وجودش قابل تامل باشه .

نتیجهء کلی : تا اون حد که دات نت امکانات و توانائی داره ، قالبلیتهای مشابهش در بستر جاوا موجوده . در برخی موارد دات نت و در برخی دیگر جاوا برتر است الا اینکه اگر کاربرد خیلی بزرگ باشه یا برنامه خاصی برای انتقال بستر وجود داشته باشه یا احتمال وجودش قوی باشه ، در هر حال " تنها گزینهء موجود" جاوا ست ، در غیر این صورت باید بررسی کرد .

</ اتمام بحث مقایسه بسترهای دات نت و جاوا >

سوال : من متوجه شدم دات نت دقیقا" چیه و جاش کجاست و متوجه شدم یک Application Server چیه و به چه دردی میخوره و باز هم متوجه شدم فرق اینها در "مقیاس" پروژه است ، حالا میخوام کمی درمورد Application Server های دیگه بدونم .

جواب : اینترنت دریائی از اطلاعات است که میتونید ازش کمک بگیرید . تجربه شخصی من به استفاده از اوراکل و اوریون محدوده . در مورد JRUN هم مطالعه کردم . اوریون یک Application Server مبتنی بر J2EE است اما برای محیط لینوکس بهینه سازی شده است . در یکی از شرکتهای نفتی ایرانی هم داره ازش استفاده میشه و فوق العاده جوابگوست . اما مثل اوراکل کامل نیست ، مثلا" بانک اطلاعاتی نداره ، باید از چیزی مثل اوراکل یا مای اسکوئل استفاده کرد ، و نواقصی از این دست اما مجانی است و سورس آزاد . www.orionserver.com .

JRUN محصول مکرومدیاست . این هم ناقصه و خیلی از قابلیتهای اوراکل رو نداره (اوراکل خیلی خیلی گرونه) اما برخی مزایای خاصش باعث میشه آدم به انتخابش فکر کنه . مثلا" قابلیت کلاسترینگ و لود بالانسینگ داره یا مثلا" ColdFusion رو حمایت میکنه و ... قیمتش هم ارزونه . مثلا" همین حالا سازمان ملی علوم و تحقیقات و تکنولوژی امریکا یعنی NIST داره از جی ران استفاده میکنه و زبان برنامه های وب اش ، خصوصا بخش امنیتی اش که زیاد کل کل میکنه هم کلد فیوژن است . اپلیکیشن سرورهای دیگه ای وجود دارن که چندان معروف نیستند . مثلا" Borland Application Server که این مورد هم مبتنی بر J2EE است و بخشی از سایت خود بورلند هم روی همین کار میکنه . اپلیکیشین سرور بورلند از اوریون و جی ران کاملتره هر چند هنوز هم از اوراکل عقب تره . دپارتمان نرم افزار "ارتش امریکا" هم بطور کامل از محصولات بورلند استفاده میکنه . برای توسعه نرم افزارهای ویندوزی از دلفی ، برای UP از توگدر و برای ارائه سرویس از اپلیکیشن سرور بورلند . (وایت پیپر هاش رو میتونید تو سایت خود بورلند پیدا کنید) و ... موارد متعدد دیگه .

جور اول : (تجربه شخصی)

کار با جاوا یا در مقیاسهای بزرگ Application Server ها جاوا واقعا" سخته . (خصوصا" اگر آدم به محیطهای قدرتمند و راحتی مثل دلفی عادت کرده باشه) در حالیکه کار با دات نت واقعا" راحته . کاربری جاوا هم مشکل تر از دات نته . در محیط دات نت اغلب تنظیمات یا وظایف کلیک اند ران هستند در حالیکه برای آماده سازی یک محیط مبتنی بر جاوا برای ارائه واقعی سرویس تخصص و تجربه لازمه و همیشه تجربه های اولیه با شکست همراه هستند . دات نت گرون نیست هر چند اگر واقع بین باشیم مجانی هم نیست . جاوا مجانی است و سورس آزاد . اون چیزی که من بهش فکر میکنم اینه که برای کاربردهای کوچک ، معمولی ، متوسط دات نت مناسبه . برای کاربردهای واقعا" بزرگ دات نت اصولا" جوابگو نیست که بخاد مناسب باشه یا نباشه و جاوا تنها گزینه است حالا میخواد خوب باشه میخواد بد باشه . یعنی اگر قرار باشه سازمانی تیم نرم افزار تشکیل بده ، یک الگو و راه حل جامع (Total Solution) برای نرم افزار انتخاب کنه ، برای برنامه نویسهاش پول خرج کنه و پول بیشتری خرج کنه تا بمونن ، قرار نیست برنامه هاش خیلی خیلی بزرگ باشن ، دات نت گزینهء خوبیه . اگر سازمانی قراره تیم نرم افزار داشته باشه و برنامه های فوق العاده بزرگ بنویسه که با توجه به نوع کاربرد احتمال تغییر پلت فرم یا خرید ماشینهای بزرگتر و قوی تر و تغییر پردازنده و .... محتمل باشه ، اون محیط مال جاواست . امیدوارم در تمام متن مطلبم به عبارت " سازمان مقیاس" که برجسته تر بود دقت کرده باشید .