



Introduction to Java EE (J2EE)

مقدمة إلى J2EE

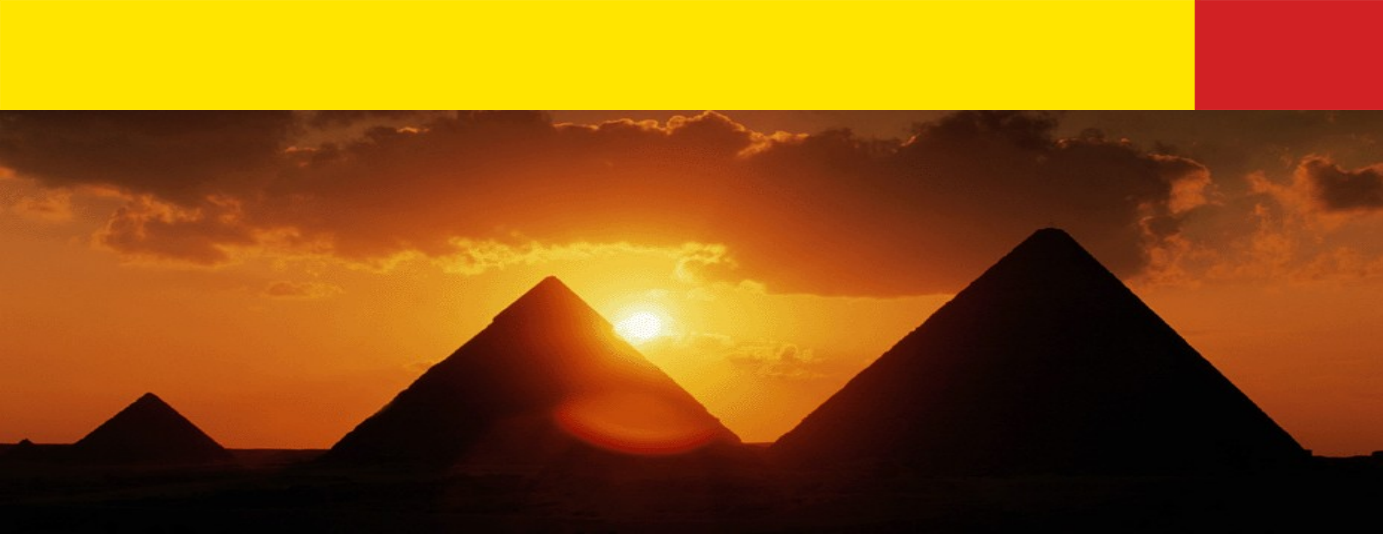
Session Objectives أهداف هذا الفصل

- Understanding the value propositions of J2EE
- Getting a **big picture** of J2EE architecture and platform أخذ فكرة شاملة عن معمارية و منصة J2EE
- Getting high-level exposure of APIs and Technologies that constitute J2EE
أخذ عرض عالي المستوى عن API و التقنيات التي تكون J2EE
 - You don't have to understand the details لست مطالب بمعرفة التفاصيل
- Understanding why J2EE is a great platform for development and deployment of web services

فهم لماذا J2EE تعتبر منصة كبيرة لبطوير و نشر خدمات الشبكة

Agenda مفكرة

- What is J2EE? لماذا J2EE
- Evolution of Enterprise Application Development Frameworks تطور منصة تطوير تطبيقات المؤسسة
- Why J2EE? لماذا J2EE
- J2EE Platform Architecture بنية منصة J2EE
- J2EE APIs and Technologies
- Standard Impl (J2EE 1.4), Compatibility Test Suite (CTS)
- BluePrints الخطط
- J2EE and Web Services خدمات الشبكة و J2EE
- How to get started كيف تبدأ



What is J2EE?

ما هي J2EE

Enterprise Computing حوسبة المؤسسة

Challenges

Portability
Diverse
Environments
Time-to-market
Core Competence
Assembly
Integration

Key Technologies

J2SE™
J2EE™
JMS
Servlet
JSP
Connector
XML
Data
Binding
XSLT

Products

App Servers
Web Servers
Components
Databases
Object to DB
tools

Legacy Systems

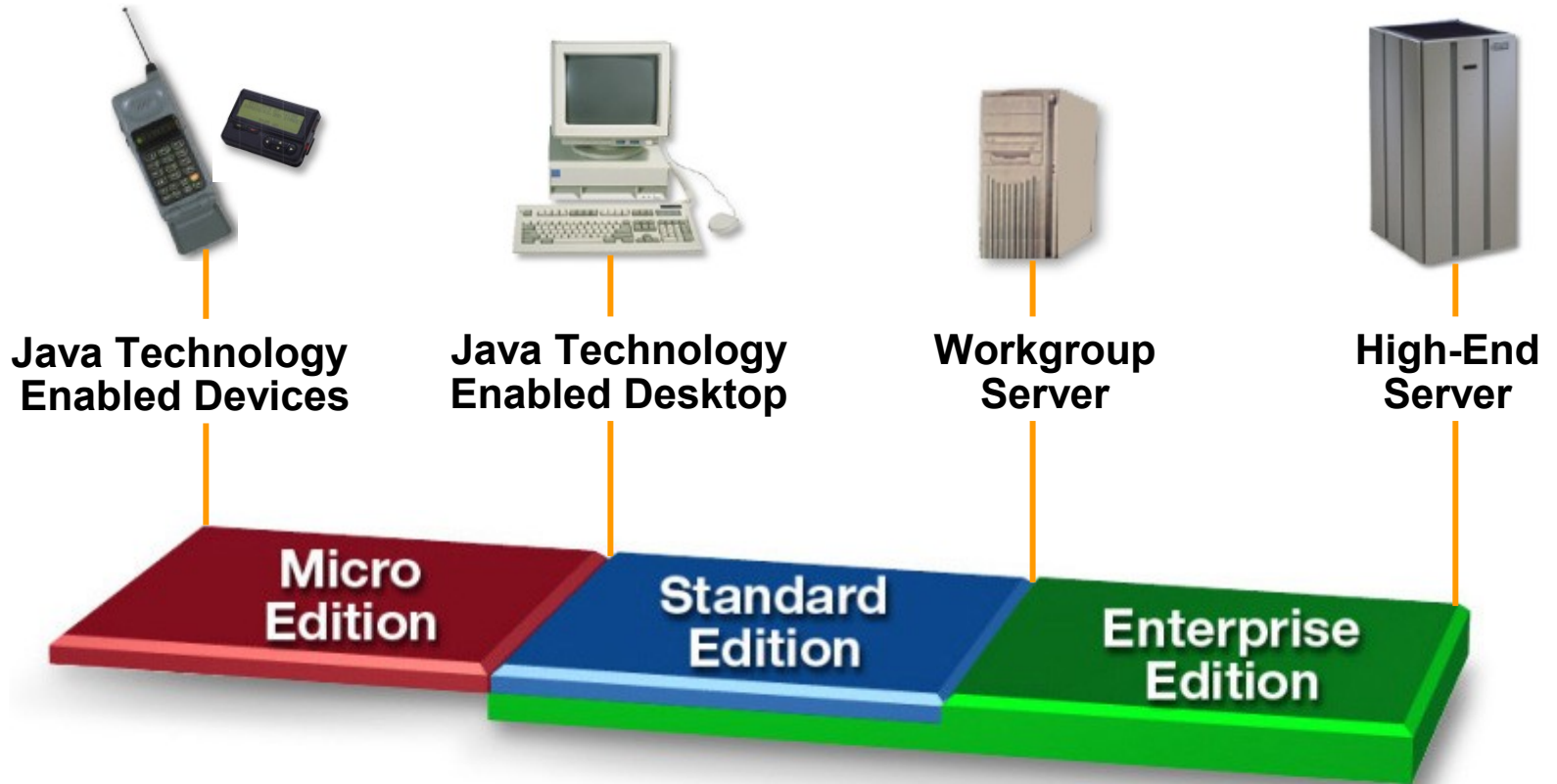
Databases
TP Monitors
EIS Systems

What Is the J2EE? ما هي J2EE

- Open and standard based platform for منصة معيارية* مفتوحة ل
- developing, deploying and managing التطوير و النشر و الإدارة
- N-tier , Web-enabled , server-centric* , and component-based enterprise applications الإتصال* بالشبكة , متعددة التطبيقات , المكونات المعتمدة على تطبيقات المؤسسة

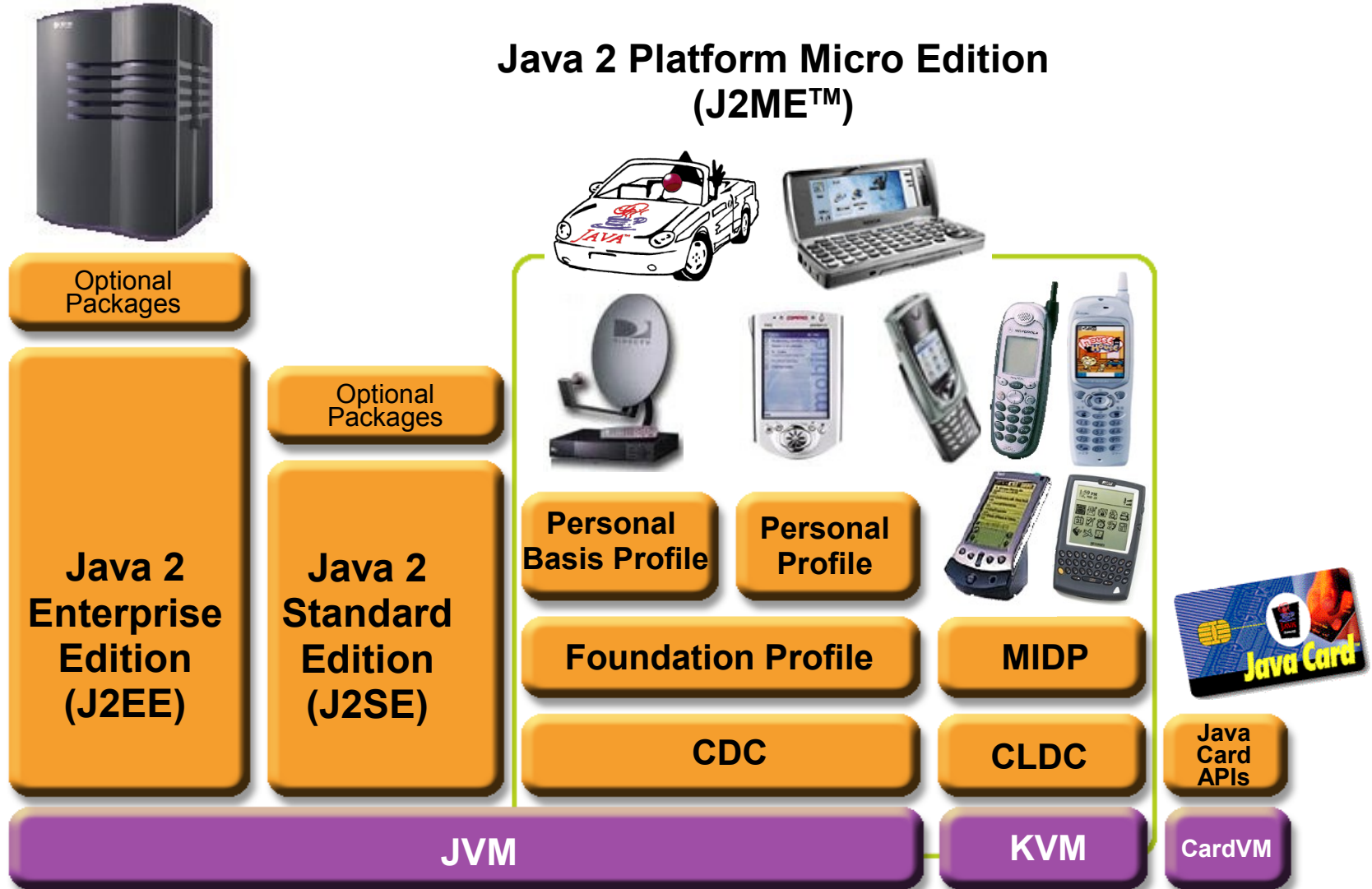
The Java™ Platform

منصة JAVA



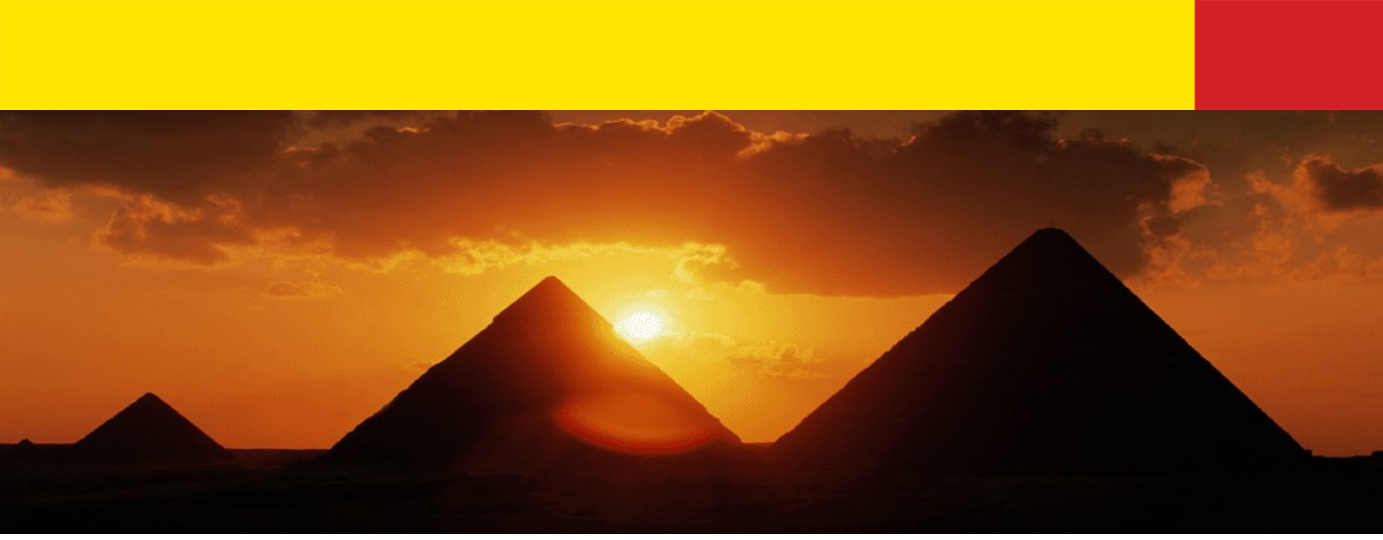
The Java™ Platform

منصة JAVA



What Makes Up J2EE? ما الذي تشكله J2EE ؟

- API and Technology specifications **API وخصائص التقنية**
- Development and Deployment Platform **منصة التطوير و النشر**
- Standard and production-quality implementation **معياري* و تطوير جودة الإنتاج**
- Compatibility Test Suite (CTS)*
- J2EE brand **علامة تجارية**
- J2EE Blueprints **خطط J2EE**
- Sample codes **عينات لمصادر**



Evolution of Enterprise Application Frameworks

تطور هيكل تطبيق المؤسسة

Evolution of Enterprise Application Framework

تطور هيكل تطبيق المؤسسة

- Single tier نموذج الطبقة الواحدة
- Two tier نموذج الطبقتان
- Three tier نموذج الثلاث الطبقات
 - RPC based بالإعتماد على RPC
 - Remote object based بالإعتماد على التحطم في الكائن عن بعد
- Three tier (HTML browser and Web server)
(متصفح HTML وخادم الشبكة)
- Proprietary application server تطبيق خادم خاص
- Standard application server تطبيق خادم معياري

About Enterprise Applications

حول تطبيقات المؤسسة

- Things that make up an enterprise application

تطبيق المؤسسة يتكون من

- Presentation logic

تقديم منطقي *

- Business logic

*

- Data access logic (and data model)

- System services

خدمات النظام

- The evolution of enterprise application framework reflects

تطور هيكل تطبيق المؤسسة يعكس

- How flexibly you want to make changes

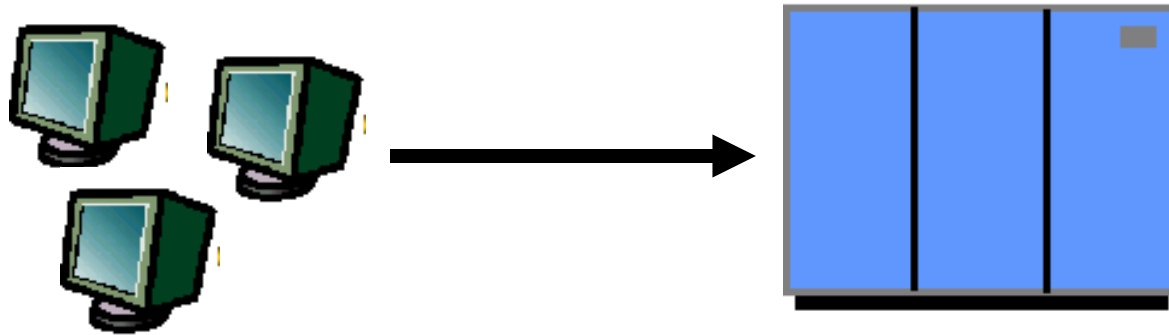
قدر المرونة في إحداث التغييرات

- Where the system services are coming from

من أين إنبثقت * خدمات النظام

Single Tier (Mainframe-based)

نموذج * الطبقة الواحدة (المعتمدة على حاسوب رئيسي)



- **Dumb terminals** are directly connected to mainframe
مطراف باتصال مع حاسوب رئيسي
- Centralized model (as opposed to distributed model)
نموذج مركزي (على عكس النموذج الموزع)
- Presentation, business logic, and data access are intertwined in one monolithic mainframe application

Single-Tier: Pros & Cons

نظام * الطبقة الواحدة (إيجابيات و سلبيات)

- Pros: إيجابيات

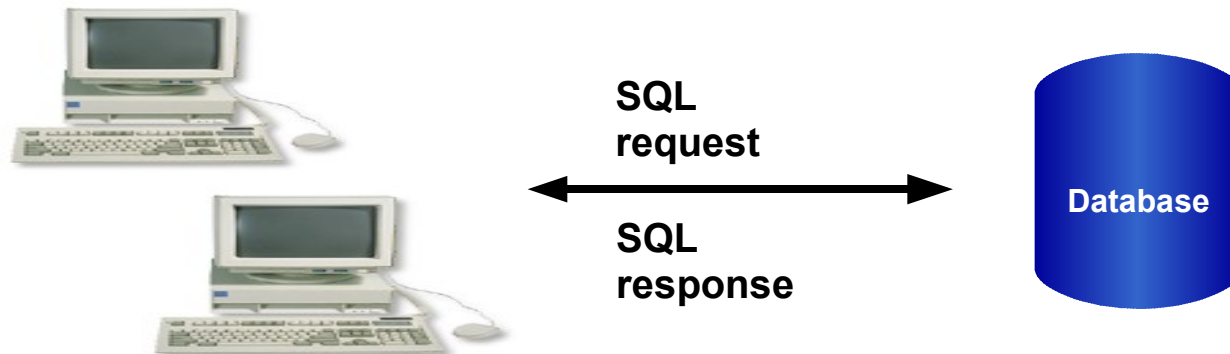
- No client side management is required
لا تتطلب صيانة من قبل العميل
- Data consistency is easy to achieve
سهولة القيام ببيانات متماسكة

- Cons: سلبيات

- Functionalityوظائف (presentation, data model, business logic) intertwined, difficult for updates and maintenance and code reuseصعوبة التحديث والصيانة وإعادة استعمال المصدر

Two-Tier

نموذج * الطبقتان



- **Fat clients talking to back end database** إتصال الحرفاء بقاعدة البيانات ثقيل جدا
 - SQL queries sent إرسال إستعلام , raw data returned إرجاع لخام البيانات
- Presentation, Business logic and Data Model processing logic in client application ... تتمركز في تطبيق الحريف

Two-Tier نموذج * الطبقتان

- Pro: إيجابيات

- DB product independence (مقارنة بنموذج الطبقة الواحدة) compared to single-tier model

- Cons: سلبيات

- Presentation, data model, business logic are intertwined (at client side), difficult for updates and maintenance (صعوبة التحديث و الصيانة)
- Data Model is “tightly coupled” (متصل بإحكام) to every client: If DB Schema changes, **all clients break** (إنقطاع جميع الحرفاء)
- Updates have to be deployed to all clients making System maintenance nightmare (يتعين أن يشمل التحديث جميع الحرفاء مما يجعل صيانة النظام كأنه كابوس)
- DB connection for every client (إتصال بقواعد البيانات لكل حريف) thus ,difficult to scale
- Raw data transferred to client for processing causes high network traffic (يبعث خام البيانات إلى الحريف لمعالجة أسباب إرتفاع تدفق * الشبكة)

Three-Tier (RPC based)

نموذج * الثلاث طبقات (المعتمدة على RPC)



- Thinner client **أنحف حريف**: business & data model separated from presentation **فصل النموذجان الأولان عن الثالث**
 - Business logic and data access logic reside in middle tier server while client handles presentation
- **Middle tier server is now required to handle system services** **خادم الطبقة المتوسطة أصبح لازماً لمعالجة خدمات النظام**
 - Concurrency control, threading, transaction, security, persistence, multiplexing, performance, etc.

Three-tier (RPC based): Pros & Cons

نموذج * الثلاث طبقات (المعتمدة على RPC)

- Pro: إيجابيات
 - Business logic can change more flexibly than 2-tier model
B.L. يمكن تغييرها بسهولة أكبر من نموذج الطبقتان
 - Most business logic reside in the middle-tier server
الكثير من B.L. تقع في خادم الطبقة المتوسطة
- Cons: سلبيات
 - Complexity is introduced in the middle-tier server
ظهور تعقيدات في خادم الطبقة المتوسطة
 - Client and middle-tier server is more tightly-coupled
(than the three-tier object based model)
متصل بإحكام أكبر
 - Code is not really reusable
المصدر حقيقة لا يصلح لإعادة الاستخدام (مقارنة ب...)
(compared to object model based ...)

Three-Tier (Remote Object based)

نموذج * الثلاث طبقات (المعتمدة على التحطم في الكائن عن بعد)



- Business logic and data model captured in objects
يمكن للكائنات من إقتناص B.L. و نموذج البيانات
 - Business logic and data model are now described in “**abstraction**” (interface language)
- Object models used: CORBA, RMI, DCOM
 - Interface language in CORBA is IDL
 - Interface language in RMI is Java interface

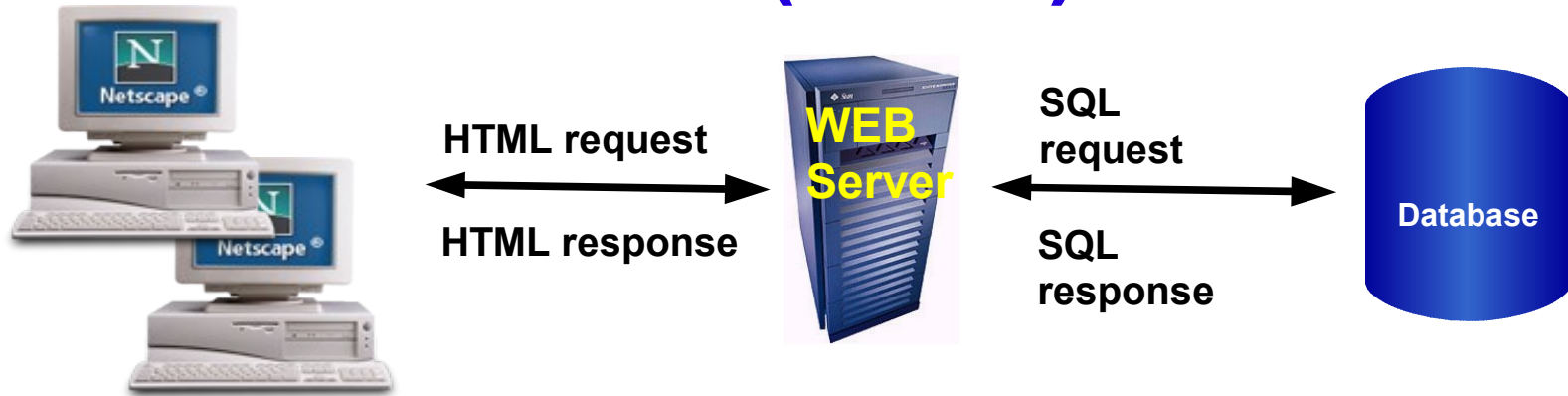
Three-tier (Remote Object based): Pros & Cons

نموذج * الثلاث طبقات (المعتمدة على التحطم في الكائن عن بعد)

- Pro: إيجابيات
 - More loosely coupled than RPC model
المزاوجة طليقة أكثر من نموذج RPC
 - Code could be more reusable يمكن إعادة إستعمال المصدر أكثر
- Cons: سلبيات
 - Complexity in the middle-tier still need to be addressed* التعقيد في الطبقة المتوسطة في حاجة أكثر إلى التوجه

Three-Tier (Web Server)

نموذج * الثلاث طبقات (خادم الشبكة)



- Browser handles presentation logic
- Browser talks Web server via HTTP protocol
المتصفح يتصل بخادم الشبكة عن طريق بروتوكول HTML
- Business logic and data model are handled by “dynamic contents generation” technologies (CGI, Servlet/JSP, ASP)

Three-tier (Web Server based): Pros & Cons

نظام * الثلاث طبقات (خادم الشبكة)

- Pro: إيجابيات
 - Ubiquitous client types
 - Zero client management
 - Support various client devices دعم للكثير من أجهزة الحريف
 - J2ME-enabled cell-phones
- Cons: سلبيات
 - Complexity in the middle-tier still need to be addressed التعقيد في الطبقة المتوسطة في حاجة أكثر إلى التوجه *

Trends إتجاهات

- Moving from single-tier or two-tier to **multi-tier** architecture الانتقال إلى البنية المتعددة الطبقات
- Moving from monolithic model to **object-based** application model الانتقال من النمط المتراس إلى نمط التطبيقات المعتمد على الكائنات
- Moving from application-based client to HTML-based client *

Single-tier vs. Multi-tier

نموذج * الطبقة الواحدة مقابل نظام المتعدد الطبقات

Single tier

نموذج الطبقة الواحدة

- No separation among presentation, business logic, database
- Hard to maintain

Multi-tier

نموذج متعدد الطبقات

- Separation among presentation, business logic, database
- More flexible to change, i.e. presentation can change without affecting other tiers

Monolithic المتراص vs. Object-based المعتمد على الكائن

Monolithic

المتراص

- 1 Binary file
- Recompiled, relinked, redeployed every time there is a change

Object-based

المعتمد على الكائن

- Pluggable parts
- Reusable
- Enables better design
- Easier update
- Implementation can be separated from interface
- Only interface is published

Outstanding Issues & Solution

المسائل القائمة و الحلول

- Complexity at the middle tier server still remains
التعقيدات في الطبقة المتوسطة لا تزال قائمة
- Duplicate system services still need to be provided for the majority of enterprise applications
لا تزال الحاجة إلى إستنساخ خدمة النظام إلى معظم تطبيق المؤسسة
 - Concurrency control, Transactions
 - Load-balancing, Security
 - Resource management, Connection pooling
- How to solve this problem? كيف نقوم بحل هذا الإشكال؟
 - **Commonly shared container** الحاويات المشتركة that handles the above system services التي تعالج خدمات النظام أعلاه
 - Proprietary versus Open-standard based

Proprietary Solution

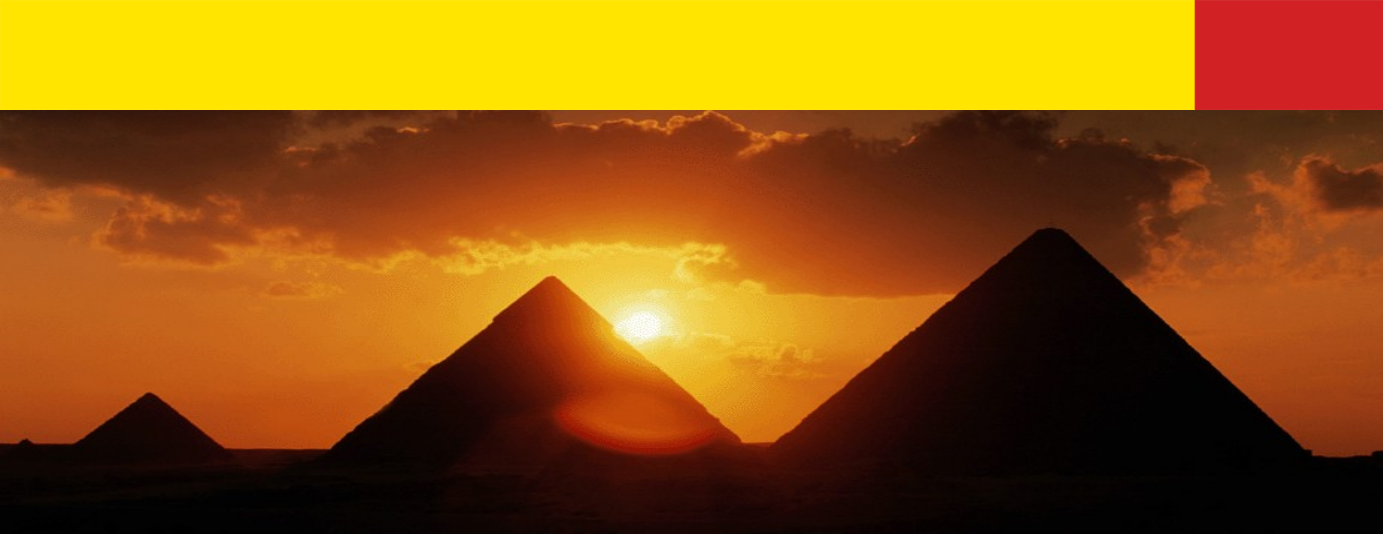
الحلول المملوكة هي الحلول الخاصة و الغير مجانية

- Use "component and container" model
 - إستخدم نمط الحاوية و المكونات
 - Components captures business logic المكونات تلتقط
 - Container provides system services الحاويات توفر خدمات النظام
- The contract between components and container is defined in a well-defined but with proprietary manner
 - العقد بين المكونات و الحاوية معرفة تعريفا جيدا و لكن على طريقة الخواص
- Problem of proprietary solution: مشكلة الحلول الخاصة:
Vendor lock-in *
- Example: Tuxedo, .NET

Open and Standard Solution

الحلول المفتوحة و المعيارية

- Use "component and container" model in which container provides system services in a **well-defined and as industry standard** إستخدم نمط الحاوية و المكونات حيث تقوم الحاوية بتوفير خدمات النظام بتعريف جيد وتحملها معيارا للصناعة
- J2EE is that standard that also provides portability of code because it is based on Java technology and standard-based Java programming APIs J2EE هو هذا المعيار الذي يقوم بتوفير قابلية لنقل المصدر لكونه مبني على تقنية جافا و على معايير جافا البرمجية API



Why J2EE?

لماذا J2EE

Platform Value to Developers

قيمة المنصة بالنسبة للمطورين

- Can use any J2EE implementation for development and deployment
يمكن إستعمال أي تطبيق J2EE للتطوير و النشر
 - Use production-quality standard implementation which is free for development/deployment
يمكن إستعمال تطبيقات معيارية لجودة الإنتاج المجانية لغرض التطوير و النشر
 - Use high-end commercial J2EE products for scalability and fault-tolerance
إستخدم منتجات J2EE التجارية القابلة للتوسع و الإستجابة للخطأ
- Vast amount of J2EE **community resources** **موارد مجتمع** **كم هائل**
 - Many J2EE related books, articles, tutorials, quality code you can use, best practice guidelines, design patterns etc.
- Can use off-the-shelf **3rd-party** business components

Platform Value to Vendors

قيمة المنصة بالنسبة للمنتجين

- Vendors work together on specifications and then **compete in implementations** المنتجون يعملون معا على تطوير المواصفات ومن ثم يعمل كل منهم لإكمال تطبيقاته
 - In the areas of Scalability, Performance, Reliability, Availability, Management and development tools, and so on في مجالات قابلية توسع البرامج و النجاعة و الثبات و أدوات التطوير و الإدارة و غير ذلك
- **Freedom to innovate** while maintaining the portability of applications حرية الإبداع مع المحافظة على القابلية لنقل التطبيقات
- Do not have create/maintain their own proprietary APIs لا يمكنك صيانة و صياغة ال API الخاصة بهم *

Platform Value to Business Customers

قيمة المنصة بالنسبة إلى الحرفاء التجاريين

- **Application portability** قابلية التطبيق لليقل
- Many implementation choices are possible based on various requirements تنفيذ العديد من الخيارات الممكنة على أساس الاحتياجات المختلفة
 - Price (free to high-end), scalability (single CPU to clustered model), reliability, performance, tools, and more
 - Best of breed of applications and platforms
- Large developer pool



J2EE APIs & Technologies

J2EE 1.4 APIs and Technologies

- J2SE 1.4 (improved)
- JAX-RPC (new)
- Web Service for J2EE
- J2EE Management
- J2EE Deployment
- JMX 1.1
- JMS 1.1
- JTA 1.0
- Servlet 2.4
- JSP 2.0
- EJB 2.1
- JAXR
- Connector 1.5
- JACC
- JAXP 1.2
- JavaMail 1.3
- JAF 1.0

Java EE 5

- JAX-WS 2.0 & JSR 181
- Java Persistence إستمرار
- EJB 3.0
- JAXB 2.0
- JavaSever Faces 1.2 – new to Platform
- JSP 2.1 – Unification w/ JSF 1.2
- StAX – Pull Parser – new to Platform

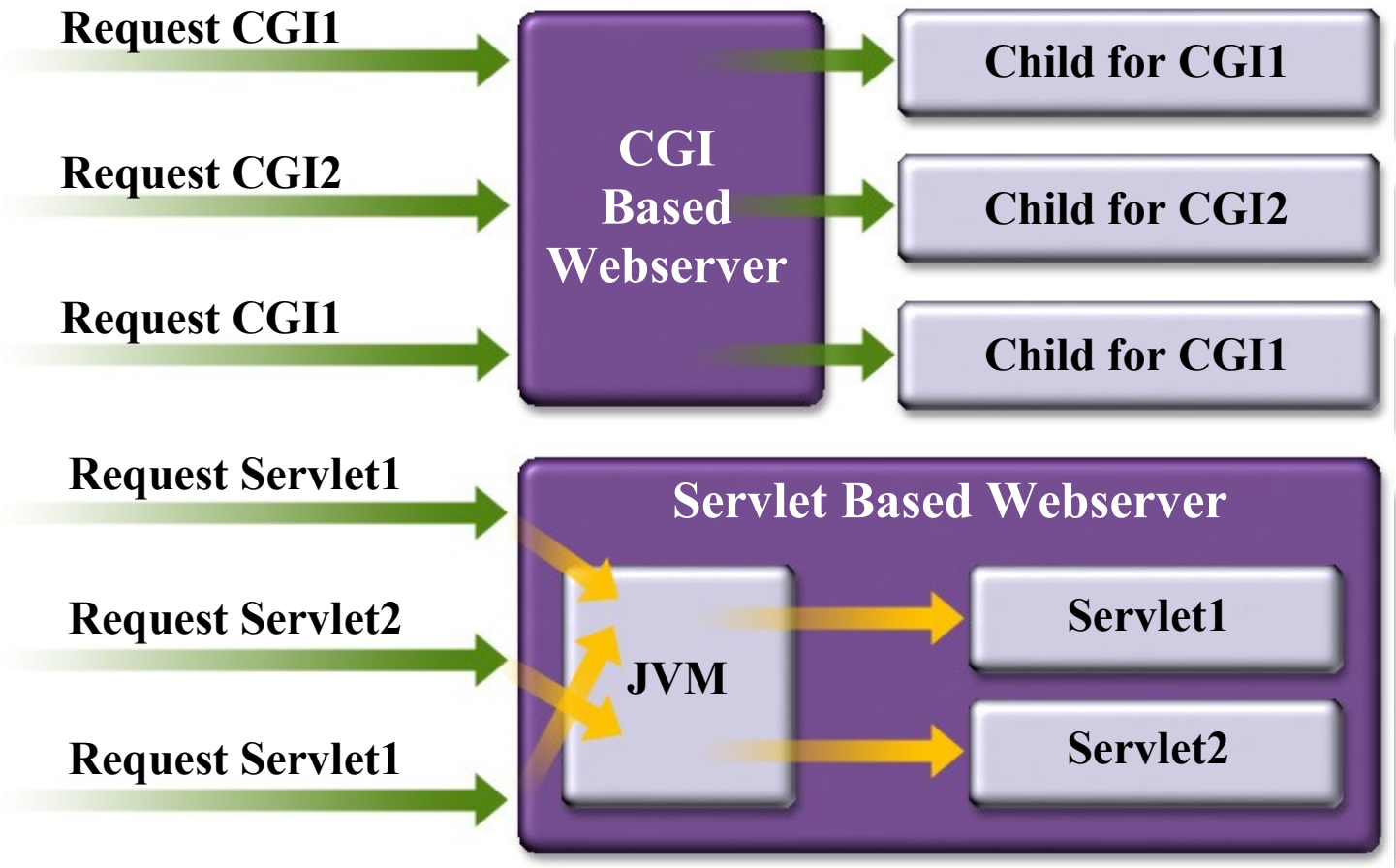


Servlet & JSP (JavaServer Pages)

What is a Servlet? ما هي Servlet

- Java™ objects which extend the functionality of a HTTP server كائنات جافا تقوم بتوسيع وظائف خادم HTTP
- **Dynamic contents generation** توليد المحتويات ديناميكيا
- Better alternative to CGI, NSAPI, ISAPI, etc. بديل أفضل
 - Efficient الكفاءة
 - Platform and server independent المنصة و استقلال الخادم
 - Session management إدارة الجلسة *
 - Java-based مبنية على جافا

Servlet vs. CGI



What is JSP Technology?

- Enables **separation** of **business logic** from **presentation** business
تمكن من الفصل بين طبقة وطبقة العرض
 - Presentation is in the form of HTML or XML/XSLT
طبقة العرض تكون على شكل
 - Business logic is implemented as **Java Beans** or **custom tags**
 - Better maintainability, reusability
قابلية للصيانة أكثر وإمكانية إعادة الاستخدام أكبر
- Extensible via custom tags
قابل للتمديد من خلال وسم مخصص
- Builds on Servlet technology
مبني بتقنية Servlet



EJB **(Enterprise Java** **Beans)**

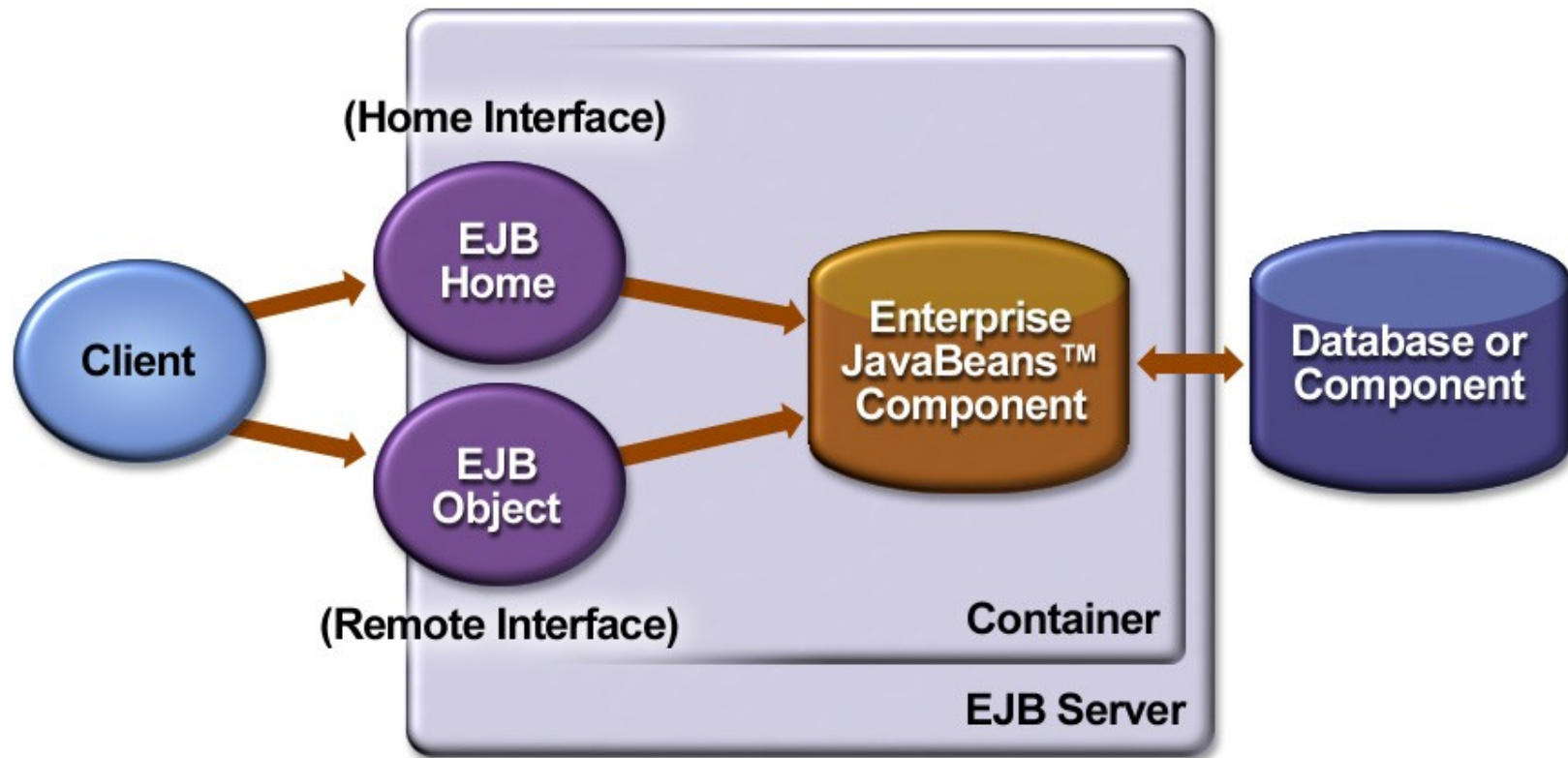
What is EJB Technology?

- A **server-side component** technology
تقنية العنصر* من جهة الخادم
- Easy development and deployment of Java technology-based application that are:
سهولة تطوير و نشر تطبيق مبني على تقنية جافا
 - Transactional, distributed, multi-tier, portable, scalable, secure, ...

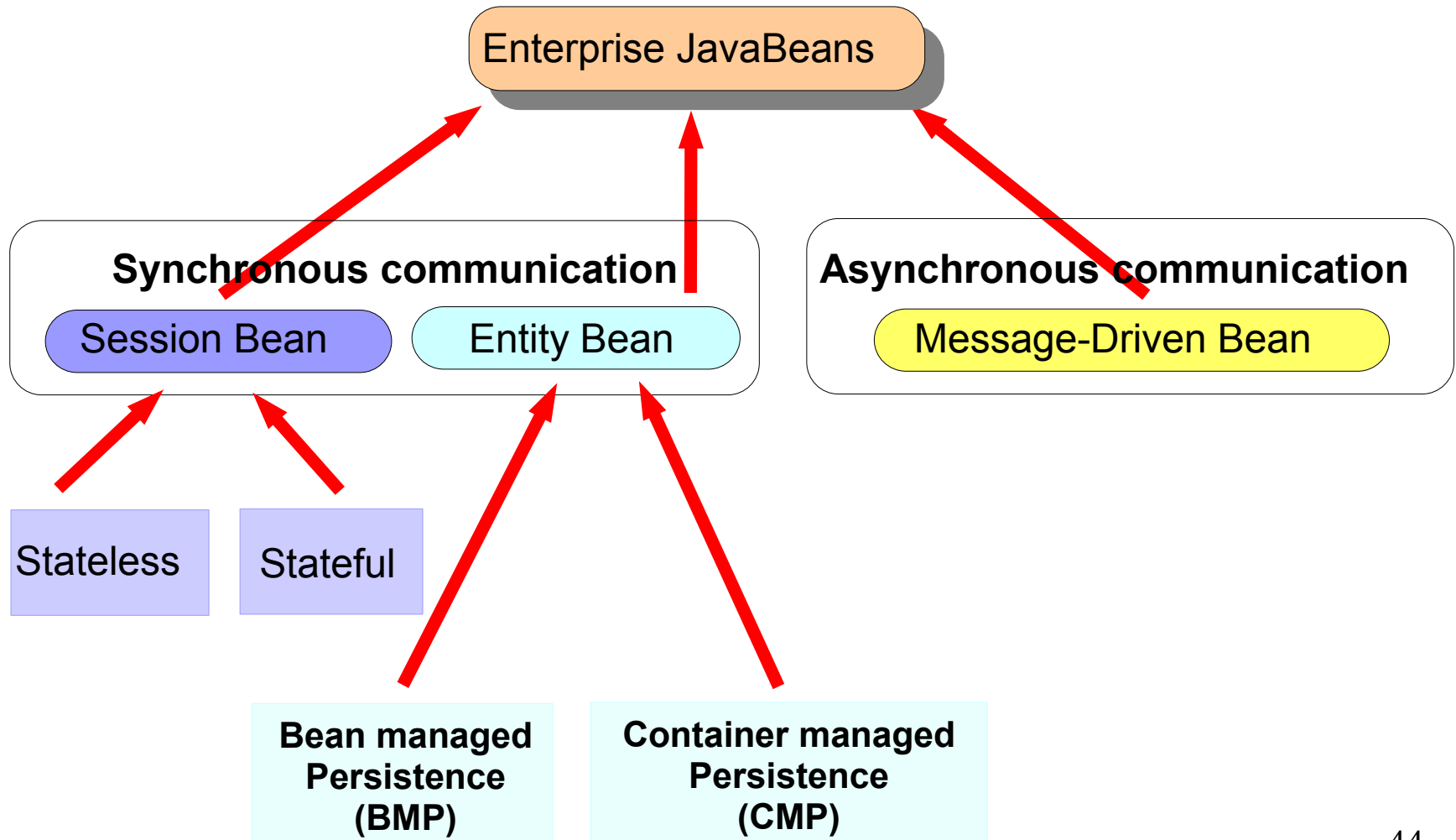
Why EJB Technology?

- Leverages the benefits of **component-model** on the server side
رفع مميزات العنصر النموذجي* إلى جهة الخادم
- Separates **business logic** from system code
 - Container provides system services
الحاوية تزود خدمات النظام
- Provides framework for **portable components**
توفر منصة للمكونات القابلة للنقل
 - Over different J2EE-compliant servers
 - Over different operational environments
- Enables **deployment-time configuration**
 - Deployment descriptor

EJB Architecture



Enterprise JavaBeans





JMS

(Java Message Service)

Java Message Service (JMS)

- Messaging systems (MOM) provide أنظمة الرسائل توفر
 - De-coupled communication
 - Asynchronous communication إتصال غير تزامني
 - Plays a role of centralized post office يلعب دور مركز البريد المركزي
- Benefits of Messaging systems مميزات أنظمة التراسل
 - Flexible مرن , Reliable ثابت , Scalable communication أنظمة الإتصال قابلة للتوسع systems
- Point-to-Point, Publish and Subscribe
- JMS defines standard Java APIs to messaging systems JMS تعرف JAVA API معيارية لأنظمة التراسل

Connector Architecture

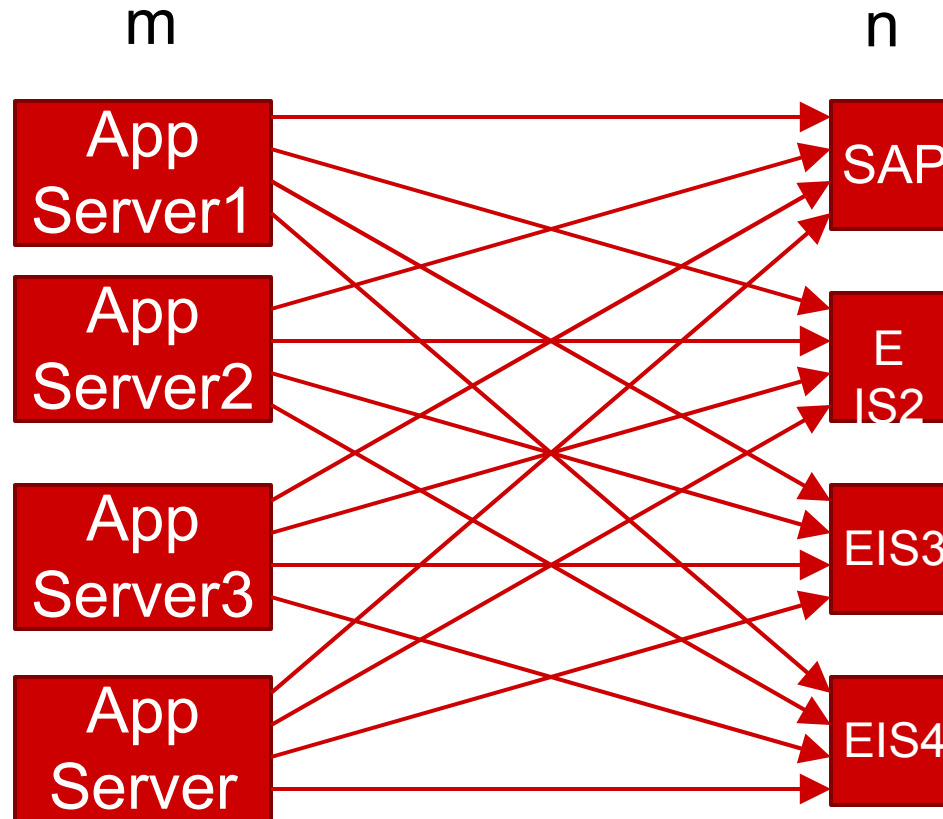
بنية الموصل*

Connector Architecture بنية الموصل*

- Defines standard API for integrating J2EE technology with **EIS** systems تعريف API معيارية لدمج تقنية J2EE مع نظام EIS
 - CICS, SAP, PeopleSoft, etc.
- Before Connector architecture, each App server has to provide an proprietary adaptor for each EIS system قبل ظهور بنية الموصل, كل تطبيق خادم كان يجب أن يوفر مهية خاص به لكل نظام EIS
 - m (# of App servers) $\times$ n (# of EIS's) Adaptors
- With Connector architecture, same adaptor works with all J2EE compliant containers مع ظهور بنية الموصل, نفس المهية يعمل مع جميع حاويات J2EE الموافقة* له
 - **1 (common to all App servers) $\times$ n (# of EIS's)** Adaptors

m x n Problem Before Connector Architecture

مشكلة m x n قبل ظهور عمارة الموصل



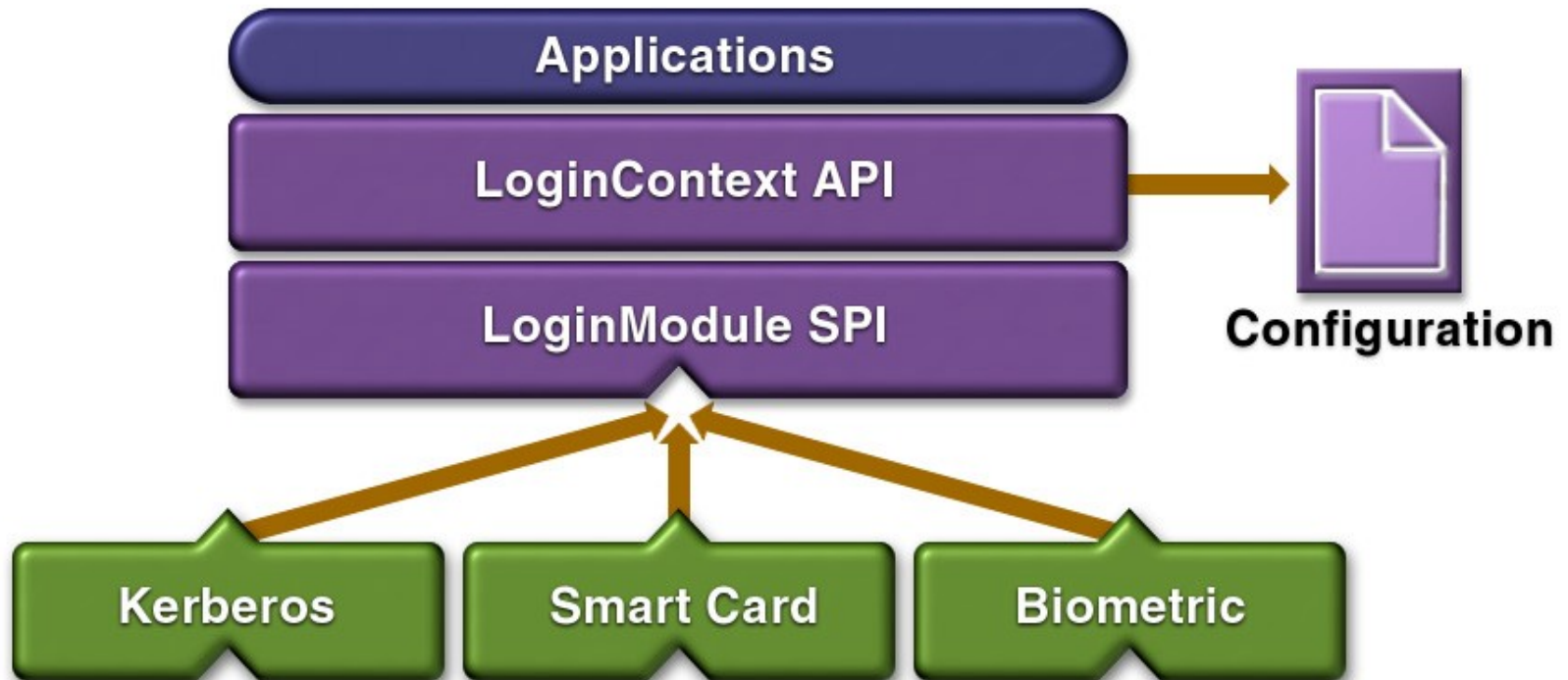
JAAS (Part of J2SE 1.4) **(Java Authentication & Authorization Service)**

إستيثاق
خدمة التصريح

JAAS: Authentication إستيثاق

- Pluggable authentication framework ملحق * منصة الإستيثاق
 - Userid/password
 - Smartcard
 - Kerberos
 - Biometric
- Application portability regardless of authentication schemes underneath
 - JAAS provides authentication scheme independent API
JAAS يوفر مخطط إستيثاق مستقل عن API
 - Authentication schemes are specified Login configuration file, which will be read by JAAS من مواصفات مخطط الإستيثاق هو التشكيلي * لعملية الولوج الذي يقوم بقراءته JAAS

JAAS Pluggable Authentication



JAAS: Authorization التصريح

- Without JAAS, Java platform security are based on قبل JAAS كانت حماية منصة جافا تعتمد على
 - Where the code originated مكان نشوء المصدر
 - Who signed the code
- The JAAS API augments this with أضافت JAAS API إلى هذا
 - Who's running the code من نفذ المصدر
- User-based authorization is now possible التصريح المعتمد على المستخدم أصبح الآن ممكنا



Other J2EE APIs & Technologies

JNDI

- Java Naming and Directory Interface
 - Utilized by J2EE applications to locate resources and objects in **portable** fashion
- يستعمل من قبل تطبيقات J2EE لتحديد موقع الموارد و الكائنات عند نقل التطبيق
- Applications use symbolic names to find object references to resources via JNDI
- التطبيقات تستعمل أسماء رمزية حتى تتمكن من إيجاد الكائن المرجعي* للموارد عن طريق JNDI
- The symbolic names and object references have to be configured by system administrator when the application is deployed.
- الأسماء الرمزية و الكائنات المرجعية يجب أن تكون قد سبق تشكيلها* من قبل مدير النظام وقت نشر التطبيق

JDBC

- Provides standard Java programming API to relational database

توفر API معيارية لقاعدة البيانات العلائقية

– Uses SQL تستعمل SQL

- Vendors provide JDBC compliant driver which can be invoked via standard Java programming API يوفر المنتجون سواق JDBC

مطابق بحيث يمكن إستدعائه من قبل API المعيارية لبرمجة جافا

J2EE Management (JSR-77)

- Management applications should be able to **discover** and **interpret** the managed data of any J2EE platform
إدارة التطبيقات يجب أن تكون قادرة على فحص* و تحليل بيانات منصة J2EE
- Single management platform can manage multiple J2EE servers from different vendors
منصة إدارة واحدة يمكنها إدارة العديد من خادمت J2EE من منتجين مختلفين
- Management protocol specifications ensure a **uniform view** by SNMP and WBEM management stations
إدارة مواصفات البروتوكولات يضمن رؤية نظامية* باستخدام إدارة المحطات SNMP و WBEM
- Leverages JMX * رفع JMX

J2EE Deployment (JSR-88) - J2EE 1.4

Tools



Standard
Deployment API
(Universal Remote)



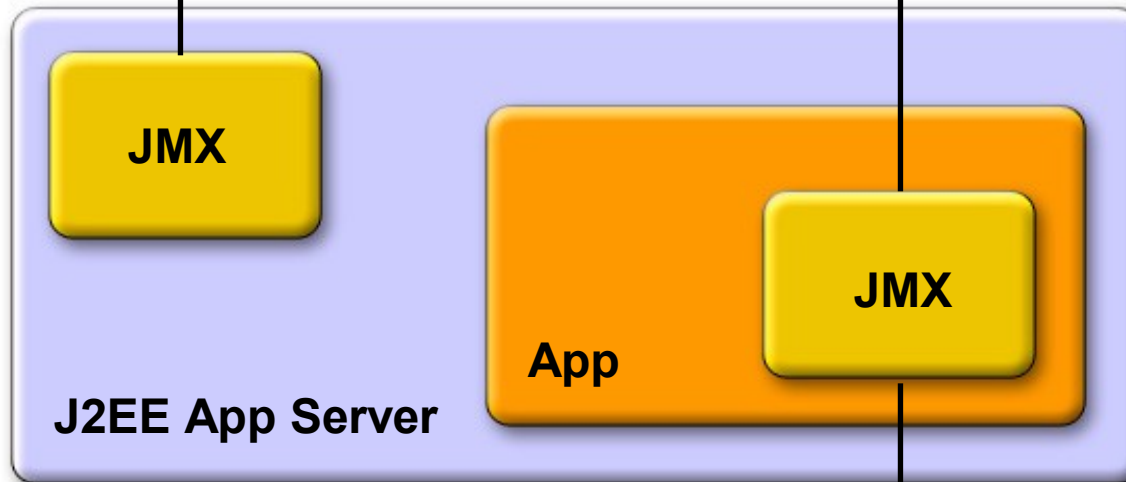
J2EE Platforms



JMX

JMX API into
the J2EE 1.4 platform

Dynamic Deployment



A single technology for the J2EE platform

JACC (Java Authorization Contract for Containers) - J2EE 1.4

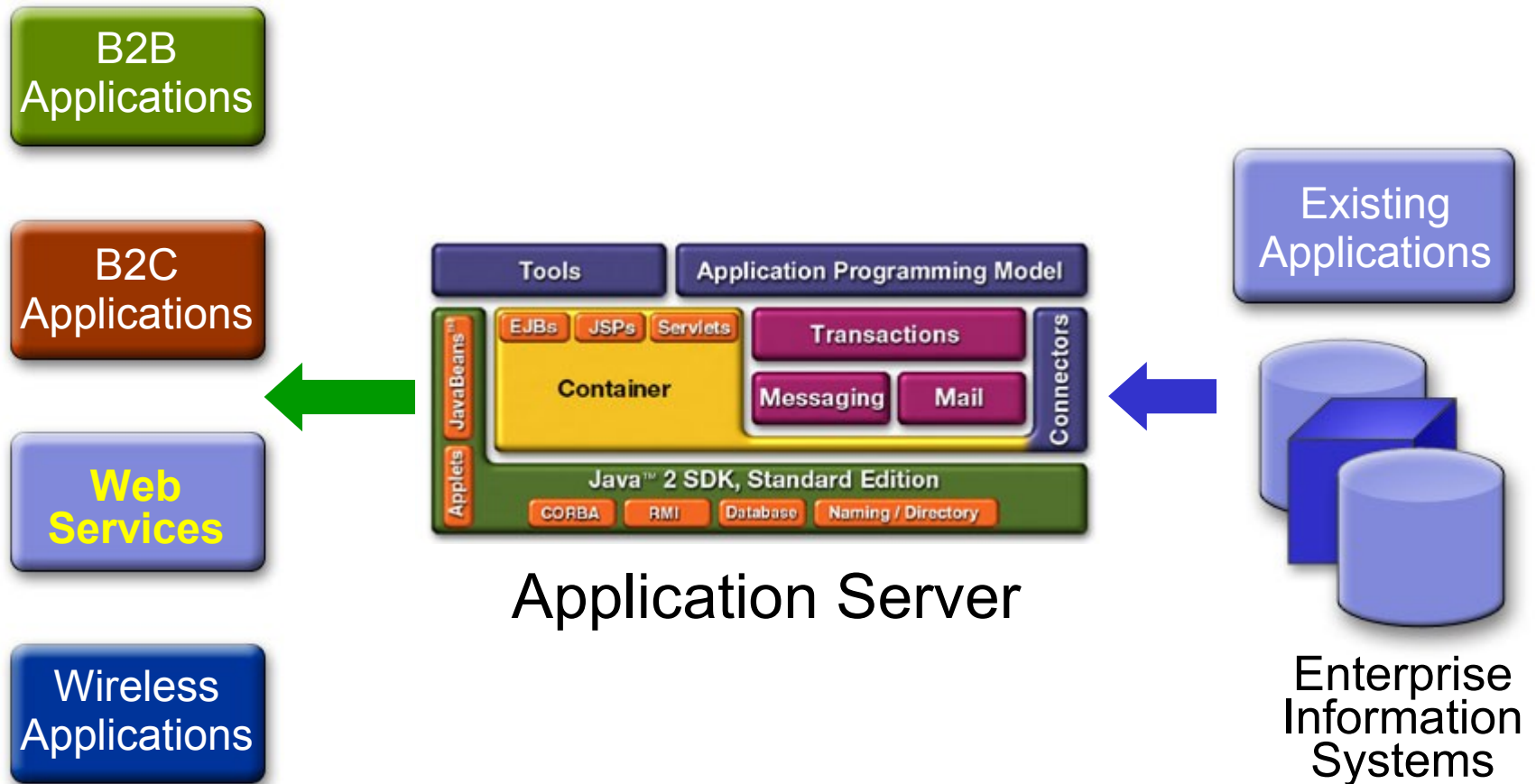
(عقد تصريح جافا للحاويات)

- Defines contract between J2EE containers and **authorization policy modules** تحديد العقود بين حاويات J2EE وسياسة* تصريح الوحدات
 - Provider configuration subcontract موفر تشكيل عقد فرعي
 - Policy configuration subcontract سياسة تكوين عقود فرعية
 - Policy enforcement subcontract سياسة تنفيذ العقود الفرعية
- Enable application servers to integrate with **enterprise user registries** and **authorization policy** infrastructure السماح للخادومات بالاندماج مع سجلات مستخدمي المؤسسة و الاندماج مع البنية التحتية لسياسة التصاريح

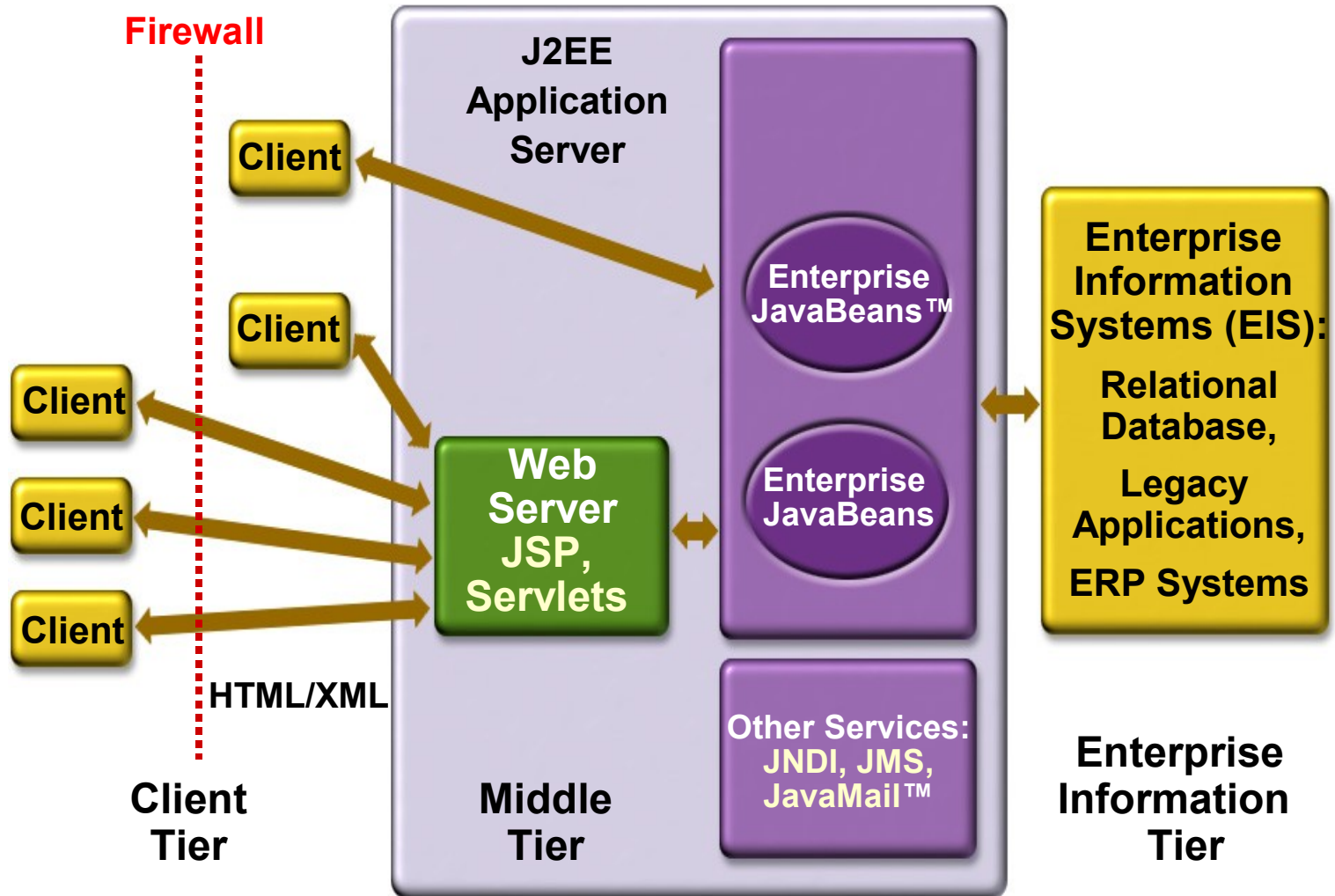


J2EE is an End-to-End Architecture

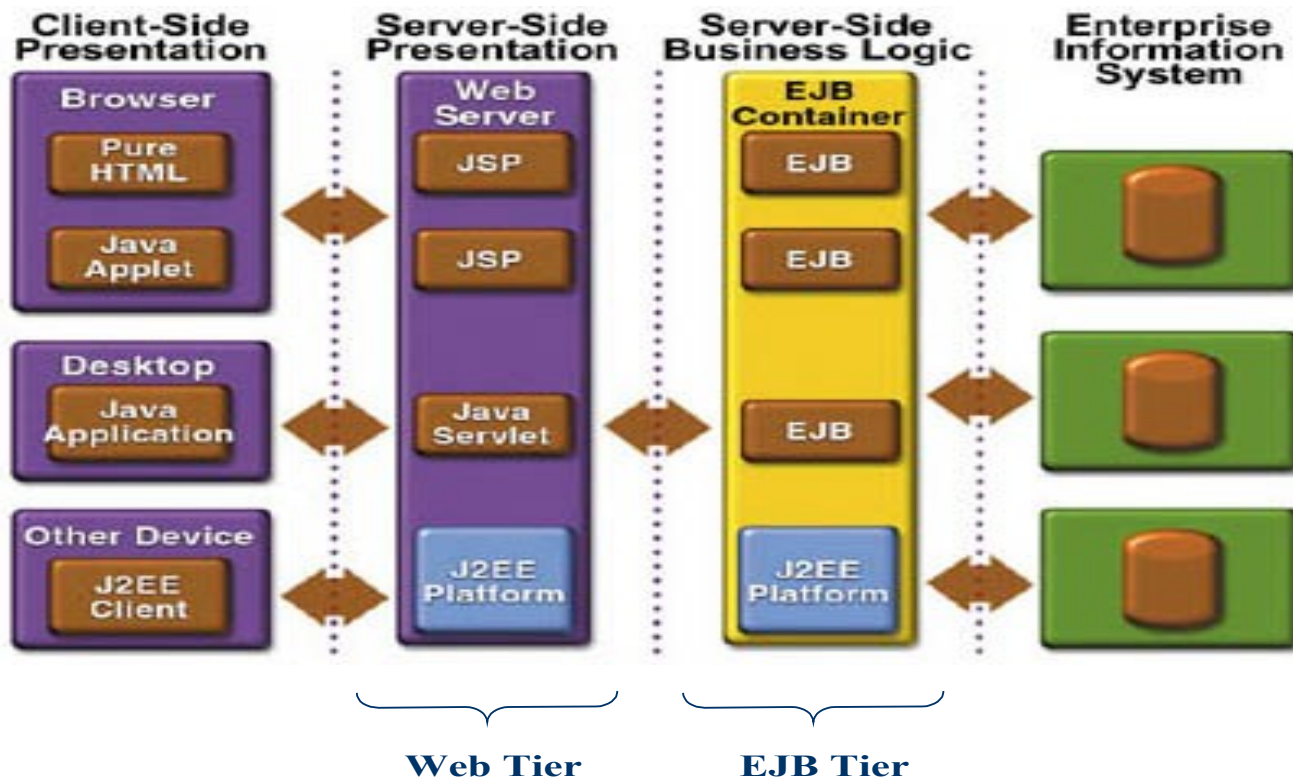
The J2EE Platform Architecture



J2EE is End-to-End Solution



N-tier J2EE Architecture





J2EE

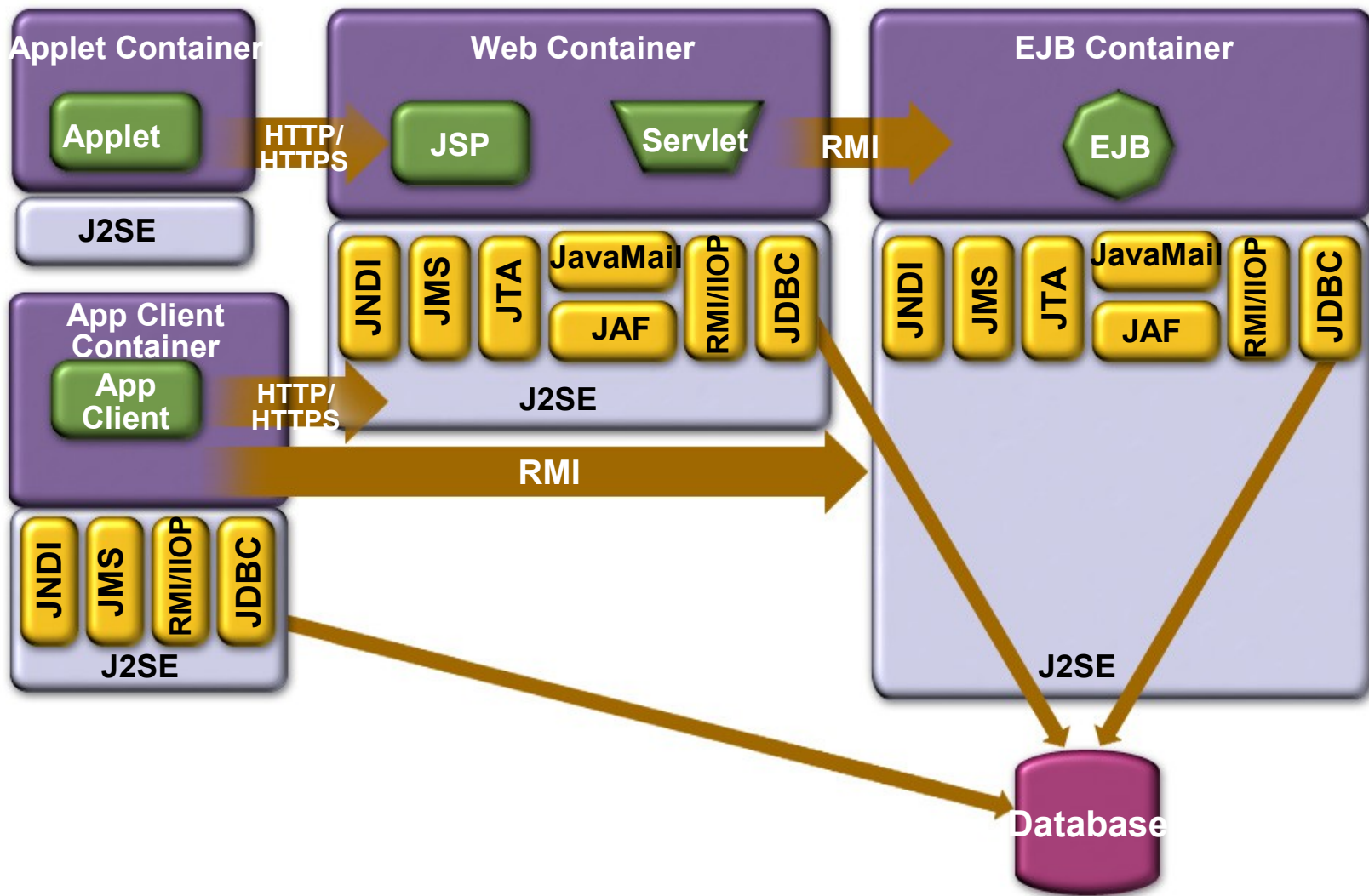
Component & Container

Architecture

بيئة المكون و الحاوية في J2EE

J2EE Containers & Components

حاويات و مكونات J2EE



Containers and Components

الحاويات و المكونات

Containers Handle

- Concurrency
- Security
- Availability
- Scalability
- Persistence
- Transaction
- Life-cycle management
- Management

Components Handle

- Presentation
- Business Logic

Containers & Components

الحاويات و المكونات

- Containers do their work invisibly
 - تقوم الحاويات بعملها بخفاء
 - No complicated APIs API غير معقدة
 - They control by interposition انها مراقبة من قبل الفاصله
- Containers implement J2EE
 - Look the same to components *
 - Vendors making the containers have great freedom to innovate



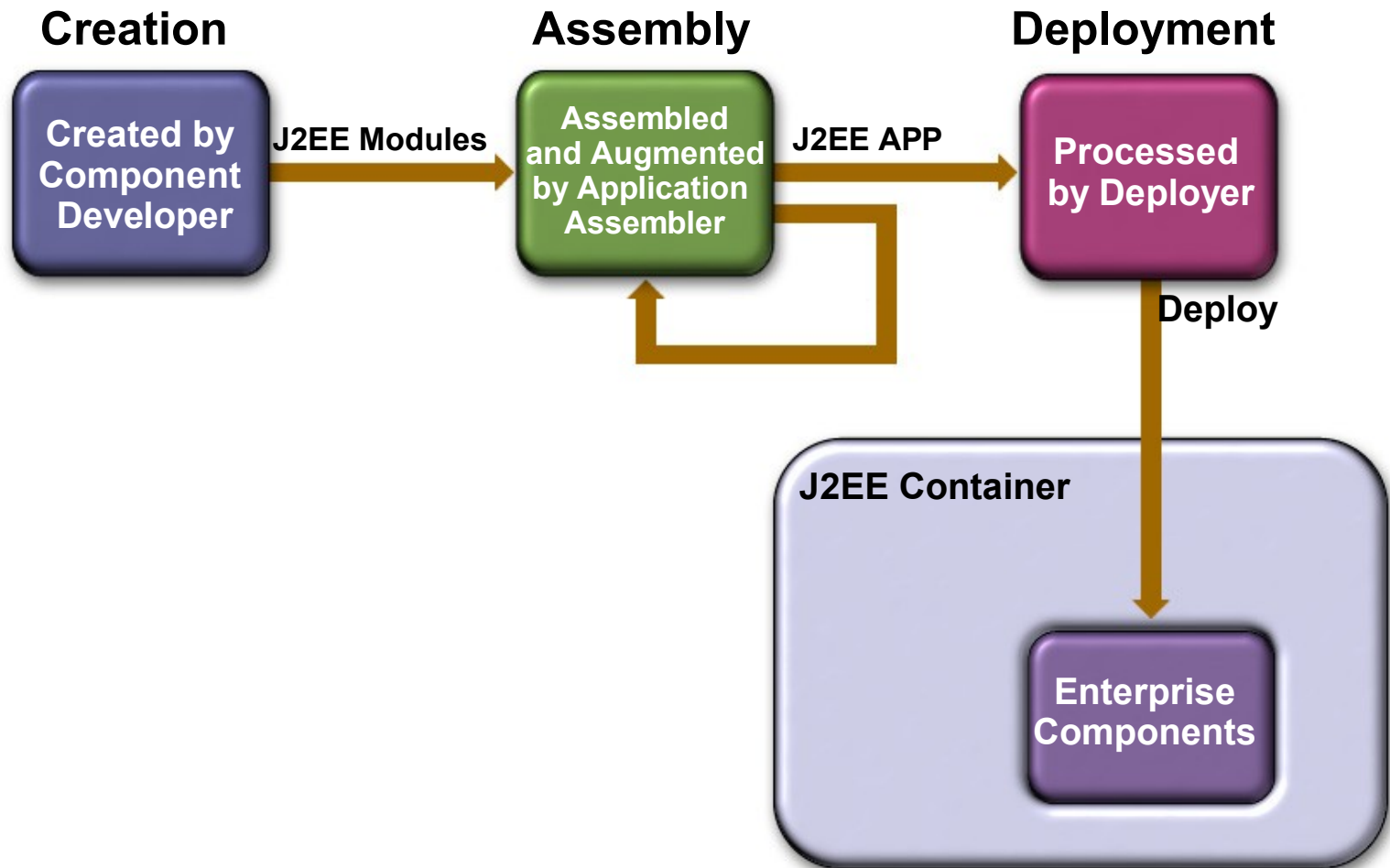
J2EE Application Development & Deployment Life Cycle

J2EE Application Development Lifecycle

دورة حياة التطوير لتطبيق J2EE

- Write and compile component code
كتابة و تجميع مكون المصدر
 - Servlet, JSP, EJB
- Write deployment descriptors for components
كتابة واصف نشر المكونات
 - From Java EE 5, you can use annotations
يمكن إستعمال الحواشي أو التعاليق إبتداء من Java EE 5
- Assemble components into ready-to-deployable package
تجميع المكونات الى حزمة جاهزة للنشر
- Deploy the package on a server
أنشر الحزمة على الخادم

Life-cycle Illustration رسم بياني لدورة الحياة



J2EE Development Roles

- Component provider مزود المكونات
 - Bean provider مزود الحبات
- Application assembler مجمع التطبيق
- Deployer الناشر
- Platform provider مزود المنصة
 - Container provider مزود الحاويات
- Tools provider مزود الأدوات البرمجية
- System administrator مدير النظام

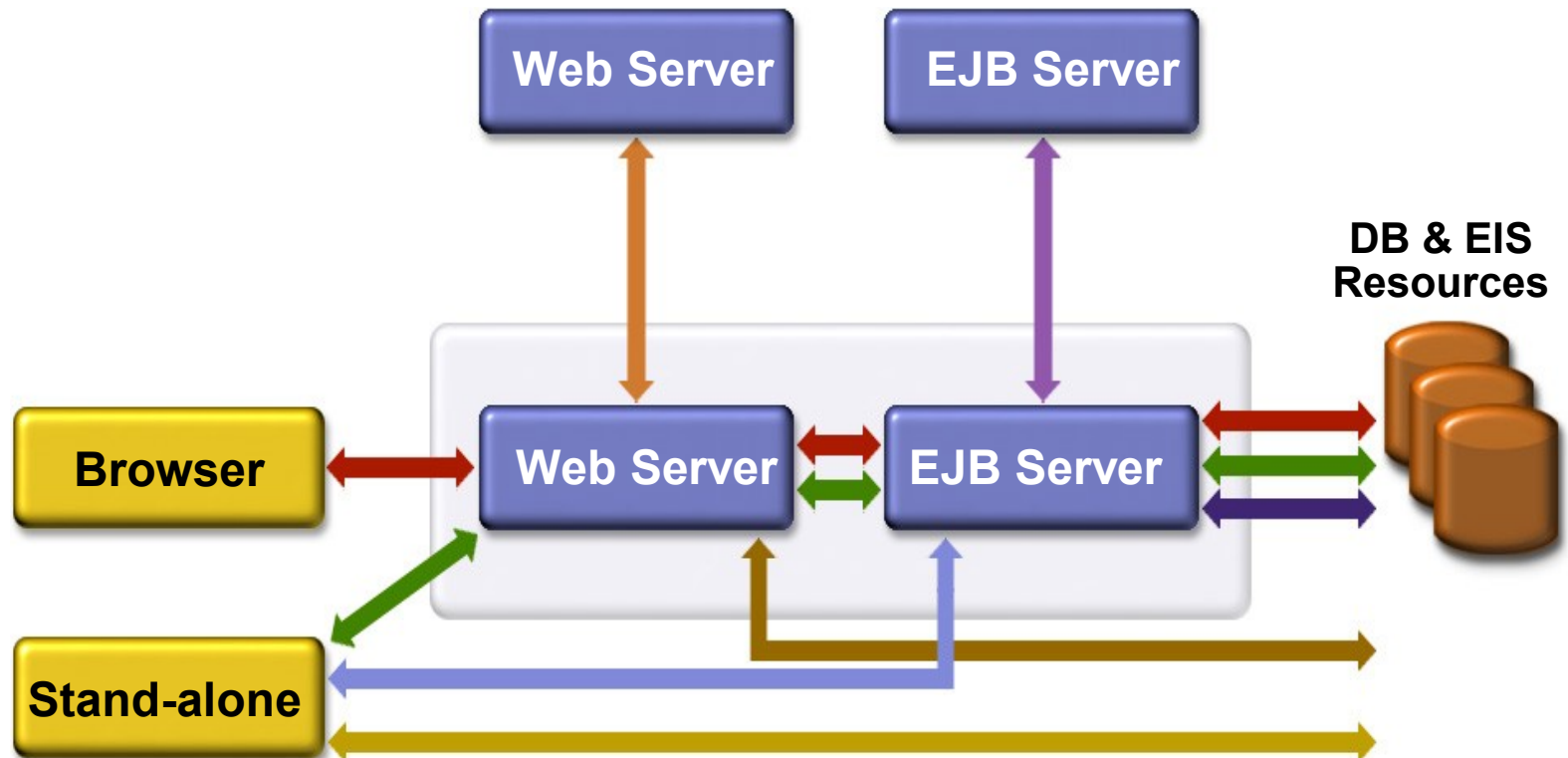
The Deployment Descriptor واصف النشر

- Gives the container instructions on how to manage and control behaviors of the J2EE components تعطي تعليمات للحاوية حتى تستطيع إدارة و التحكم في تصرفات مكونات J2EE
 - Transaction
 - Security حماية
 - Persistence إستمرار
- Allows **declarative** customization تسمح * بالتخصيص (as opposed to programming customization التقريري) (على عكس * التخصيص البرمجي)
 - XML file
- Enables **portability** of code إمكانية نقل المصدر



J2EE Application Anatomies

Possible J2EE Application Anatomies



J2EE Application Anatomies

- 4-tier J2EE applications
 - HTML client, JSP/Servlets, EJB, JDBC/Connector
- 3-tier J2EE applications
 - HTML client, JSP/Servlets, JDBC
- 3-tier J2EE applications
 - EJB standalone applications **تطبيق قائم بذاته**, EJB, JDBC/Connector
- B2B Enterprise applications
 - J2EE platform to J2EE platform through the exchange of JMS or XML-based messages **من منصة J2EE إلى منصة**
J2EE من خلال تبادل رسائل مبنية على تقنية JMS أو XML

Which One to Use? أي نوع من هذه التطبيقات نستعمل

- Depends on several factors يعتمد ذلك على العديد من العوامل
 - Requirements of applications متطلبات التطبيق
 - Availability of EJB tier توفر طبقة EJB
 - Availability of developer resource *توفر موارد المطورين



J2EE 1.4

Standard Implementation, Compatibility Suite, Brand

Standard Implementation

- Under J2EE 1.4 SDK, it is Sun Java Application Server Platform Edition 8
- Production-quality J2EE 1.4 compliant app server
- Free to develop and free to deploy حر في التطوير
و حر في النشر
- Seamless upgrade path to Sun Java Application Server Enterprise Edition

Compatibility Test Suite (CTS)

- Ultimate Java™ technology mission:
 - Write Once ^{إكتب واحدة}, Run Anywhere™ ^{شغل أينما كنت}
 - My Java-based application runs on any compatible Java virtual machines ^{تطبيقات الجافا الخاصة بي أطبقها على أي JVM موافق}
 - My J2EE based technology-based application will run on any J2EE based Compatible platforms ^{تطبيق J2EE الخاص بي أشغله على أي منصة J2EE موافقة}

J2EE Application Verification Kit (J2EE AVK)

حقيبة مراجعة تطبيقات J2EE

- How can I test my J2EE application portability? كيف يمكنني تجربة قابلية تطبيقي على النقل
 - Obtain the J2EE RI 1.3.1 and the **J2EE Application Verification Kit (J2EE AVK)**
- Self verification of application مراجعة ذاتية للتطبيق
 - Static verification
 - Dynamic verification
- Obtain the tests results , verify that all criteria are met تحصل على نتائج التجارب , تحقق من وجود جميع المواصفات

Compatible Products for the J2EE Platform (Brand)

المنتجات المتوافقة مع منصة J2EE (العلامات التجارية)

ATG

Bea Systems

Borland

Computer
Associates

Fujitsu

Hitachi

HP

IBM

IONA

iPlanet

Macromedia

NEC

Oracle

Pramati

SilverStream

Sybase

Talarian

Trifork



**Java Compatible™
Enterprise Edition**

The J2EE Platform “Ecosystem,” Application Servers and...

- Tools **الأدوات البرمجية**
 - IDE's: Borland JBuilder Enterprise, WebGain Visual Cafe', IBM Visual Age for Java™, Forte™ for Java™, Oracle JDeveloper, Macromedia Kawa
 - Modeling, Performance, Testing, etc.
- Enterprise Integration: Connectors, Java Message Service (JMS) API, XML
- Components **المكونات**
- Frameworks **المنصات**
- Applications **التطبيقات**

Major Investment in Compatibility by the Industry

* أعظم إستثمار في التوافق مع الصناعة

- Sun has spent scores of engineer years developing tests
أمضت شركة SUN عشرات السنوات في التطوير و الإختبار
- Licensees have spent scores of engineer years passing the tests
المرخصون * أمضوا عشرات السنين يجرون الإختبارات
- Testing investment on top of specification investment, implementation investment, business investments
الإستثمار في الإختبارات كان مقدم على الإستثمارات الأخرى
- In total, tens of millions of dollars invested in J2EE platform compatibility by the industry
في المجمال تم إستثمار
84 * عشرات الملايين من الدولارات لتتوافق منصة J2EE مع الصناعة



J2EE Blueprint ^{مخطط} & Pet Store Application

J2EE Blueprint

مخطط J2EE

- Best practice guidelines , design patterns , توجيهات لأفضل الممارسات , مبادئ التصميم and design principles تصميم أنماط
 - MVC pattern
- Covers all tiers تغطية جميع الطبقات
 - Client tier طبقة الحريف
 - Web tier طبقة الشبكة
 - Business logic (EJB) tier
 - Database access tier طبقة المفاز إلى البيانات
- Sample codes come with J2EE 1.4 SDK عينات لبعض المصادر التي وزعت مع J2EE 1.4 SDK
 - Java Pet Store, Adventure builder



Why J2EE for Web Services?

لماذا نستخدم J2EE في خدمات الشبكة

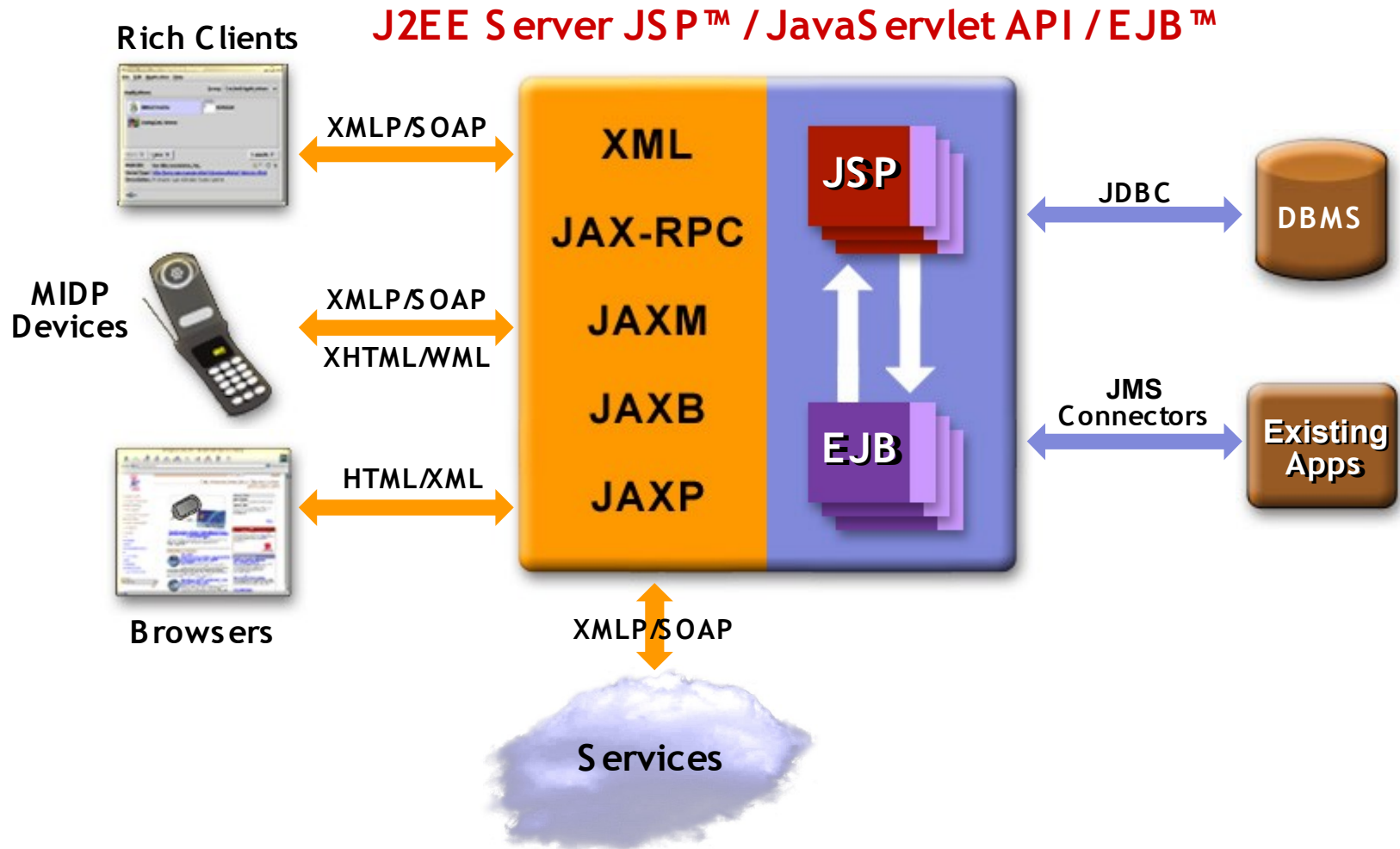
Why J2EE for Web Services?

لماذا نستخدم J2EE في خدمات الشبكة

- Web services is just **one of many service delivery channels** of J2EE
خدمة توصيل قنوية من واحد إلى مجموعة
- No architectural change is required لا يستلزم تغيير في البنية
- Existing J2EE components can be easily exposed as Web services
مكونات J2EE الحالية يمكن أن تعرض بسهولة كخدمات شبكة
- Many **benefits** of J2EE are **preserved** for Web services
فوائد كثيرة قدمتها J2EE لخدمات الشبكة
- Portability , Scalability , Reliability
الثبات , القابلية للتوسع , القابلية للنقل
- No single-vendor lock-in

Web Services Model Over J2EE

نموذج لخدمات شبكة من خلال J2EE



Where Are We Now? أين نحن الآن؟

- **Java APIs** for Web Services are being developed very rapidly
تطورت **API** الخاصة بخدمات الشبكة سريعا
- Web services support on WUST (WSDL, UDDI, SOAP) ready now **WUST** جاهزة الآن
خدمات الشبكة التي تعتمد على **WUST** جاهزة الآن
- Next layer Web services work in progress
طبقة أخرى لخدمة الشبكة هي في طور الإنجاز
- **Tools** are available now for exposing existing J2EE components as Web services
خدمات برمجية جاهزة
الآن لعرض مكونات **J2EE** الحالية كخدمات شبكة
- J2EE community has defined **overall framework** for Web Services (J2EE 1.4, Web services for J2EE)
حدد مجتمع **J2EE** الإطار العام لخدمات الشبكة

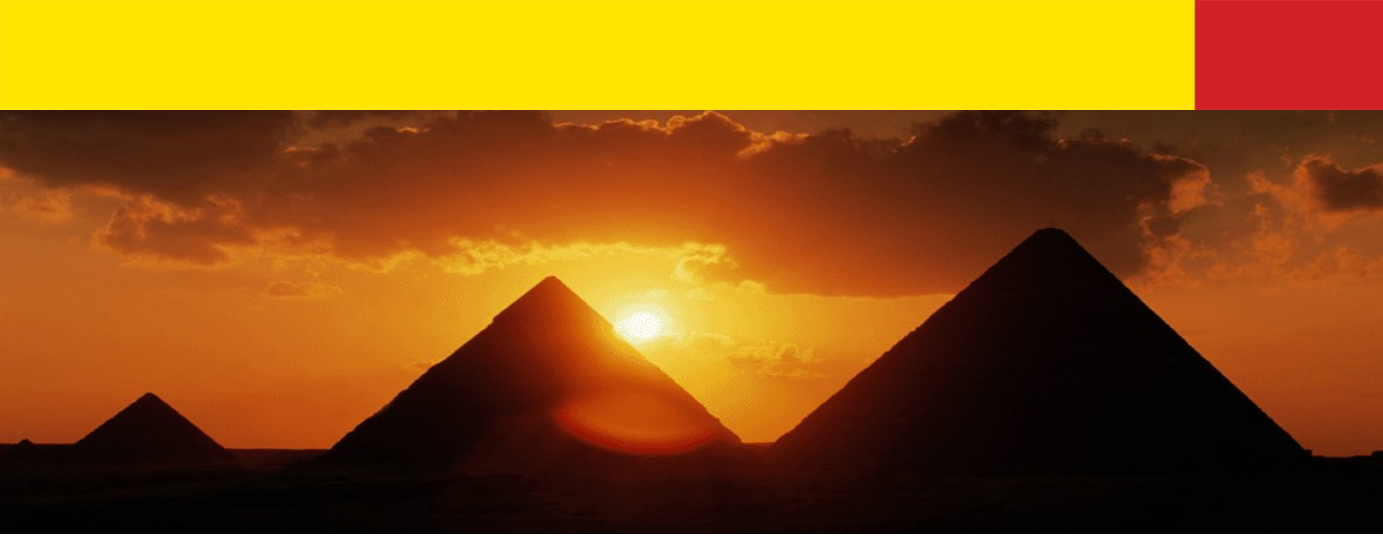
Design Goals J2EE 1.4 Web Services Framework

أهداف تصميم منصة خدمات الشبكة J2EE 1.4

- **Portability** of Web services component
 - قابلية مكونات خدمات الشبكة للنقل
 - Over different vendor platform من خلال صانعي منصات مختلفة
 - Over different operational environment
- Leveraging existing J2EE **programming models** for service implementation
 - رفع نماذج J2EE البرمجية الحالية للإستعمال من قبل خدمات الشبكة
- **Easy** to program and deploy سهولة في البرمجة و النشر
 - High-level Java APIs عالية المستوى
 - Use existing deployment model إستعمال نموذج النشر الحالي

J2EE 1.4 Web Services Framework

- J2EE 1.4 (JSR 151)
- Web services for J2EE (JSR 109)
- JAX-RPC (JSR 101)
- JAXR (Java API for XML Registries)
- SAAJ (SOAP with Attachments API for Java)
- EJB 2.1



How to Get Started

كيف نبدأ

Step1: For Beginners and Intermediate J2EE Programmers

الخطوة الأولى بالنسبة للمبرمجين المبتدئين و المتوسطين

- Follow along with this course تابع معنا من خلال هذه الدورة
- Start using J2EE IDE of your choice إبدأ باستعمال بيئة تطوير متكاملة من إختيارك
- Try open source IDE's جرب المفتوحة المصدر منها
 - NetBeans IDE 5.0 (netbeans.org)
 - Excellent out of the box J2EE support
 - We will use NetBeans IDE 5.0 or NetBeans IDE 5.5 as our default IDE in this course سنعتمد NetBeans IDE 5.0 أو NetBeans IDE 5.5 كبيئة تطوير متكاملة في هذه الدورة
 - Lots of tutorials توفر الكثير من الشروحات
 - Eclipse

Step2: Next Step (For Advanced J2EE Programmers)

الخطوة الثانية بالنسبة للمبرمجين المحترفين

- Learn practical open-source solutions

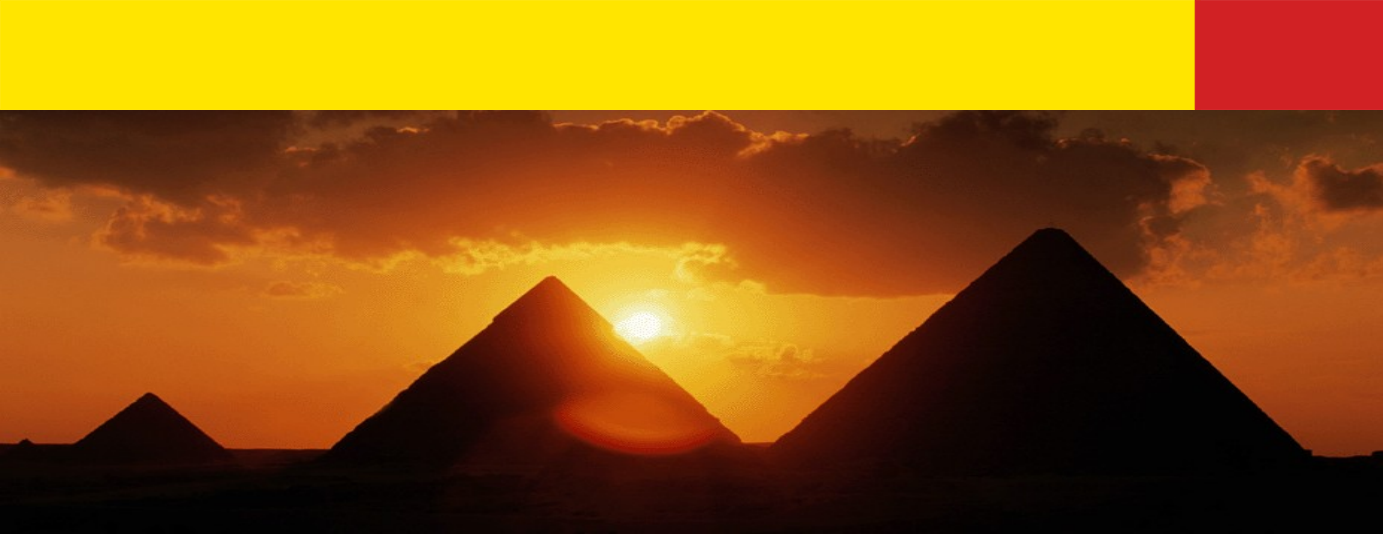
تعلم الحلول العملية المفتوحة المصدر

- Spring framework (for light-weight framework)
- Hibernate (for O/R mapping)
- JDO (for transparent persistence)
- Struts, WebWork, Tapestry (for Web-tier frameworks)
- JUnit (for unit testing)
- Log4j (for logging)
- Many more

Step3: Next Step (For Advanced J2EE Programmers)

الخطوة الثالثة بالنسبة للمبرمجين المحترفين

- There is no shortage of quality J2EE online resources ليس هناك نقص في جودة موارد J2EE
 - java.sun.com/j2ee
 - www.theserverside.com
 - www.javapassion.com/j2ee/J2EEresources.html#J2EEResourceSites



Summary & Resources

الحوصلة و المراجع

Summary حوصلة

- J2EE is the platform of choice for development and deployment of n-tier, web-based, transactional, component-based enterprise applications
يمكن إختيار منصة J2EE لتطوير و نشر تطبيقات المؤسسة المتعددة الطبقات ...
- J2EE is standard-based architecture
بنية J2EE المعيارية *
- J2EE is all about community
- J2EE evolves according to the needs of the industry
J2EE يتطور وفقا لاحتياجات الصناعة

Resources المراجع

- J2EE Home page
 - java.sun.com/j2ee
- J2EE 1.4 SDK
 - java.sun.com/j2ee/1.4/download.html#appserv
- J2EE 1.4 Tutorial
 - java.sun.com/j2ee/1.4/download.html#appserv
- J2EE Blueprints
 - java.sun.com/blueprints/enterprise/index.html

NetBeans Resources

Netbeans مراجع

- NetBeans IDE Homesite
 - <http://www.netbeans.org>
- NetBeans IDE Tutorials/Articles Master index
 - <http://www.javapassion.com/netbeans/masterindex.html>



Passion!