# البنيان المؤسساتي
# Enterprise Architecture (EA)

## ITIS411 >> LEC 7#

د.حنان الداقيز

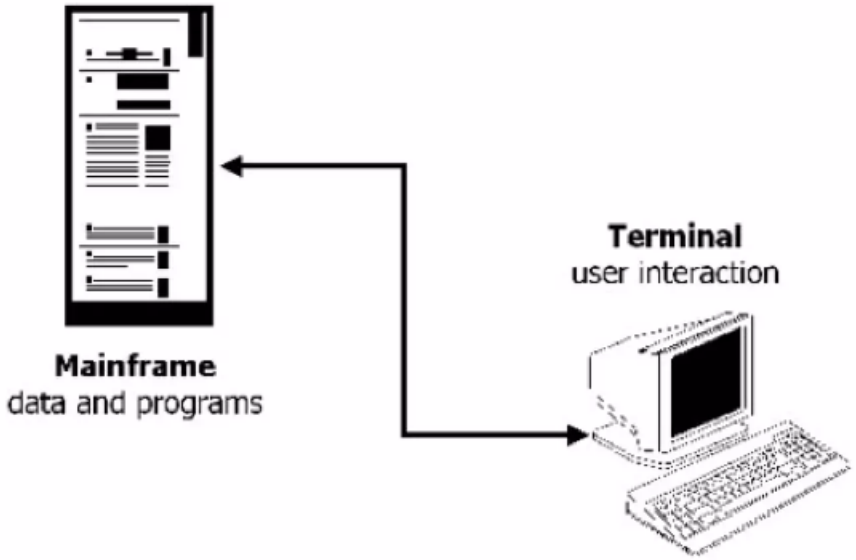
# الاهداف

# Service Oriented Architecture SOA
# معمارية الخدمية الموجهة

# تاريخ و هيكلية البرامج

## المرحلة الأولي : Tier1-Mainframe

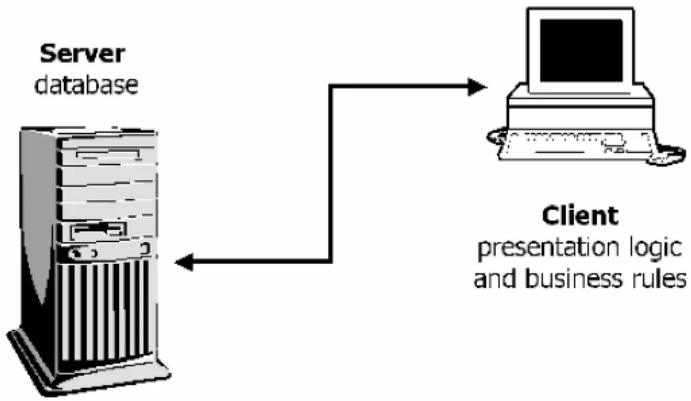
من المتعارف عليه بداية أنه يوجد مجموعة من البرامج و قواعد البيانات التى تعمل على أجهزة الحاسب الآلى و التى من الممكن لمستخدمى هذه البرامج التعامل معها عن طريق الاجهزة الموصل عليها هذه البرامج و قواعد البيانات. اي ان الفكرة التي فرضت نفسها آنذاك هي عمل كمبيونر واحد به كلا من البرامج وقاعدة البيانات وهو بالضبط ما كان في Mainframe والذي يسمى بـ Tier1 او الطبقة الاولى.



# تاريخ و هيكلية البرامج

**المرحلة الثانية : Tier2-Client Server**

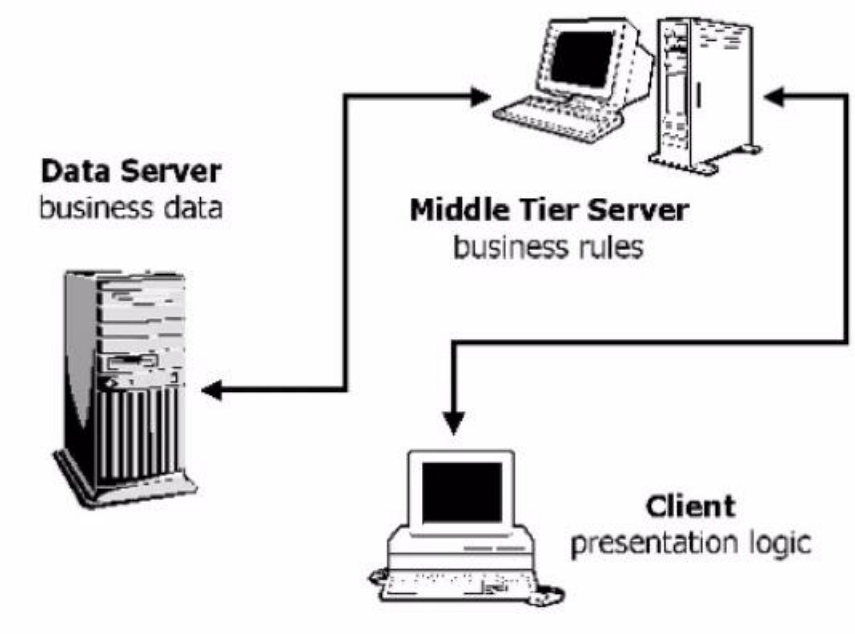
- جاءت فكرة ال Tier2 لتخفيض العبئ علي الكمبيوتر ( وجود البرامج و قاعدة البيانات في طبقة واحدة و عمل الأعمال المنطقية ) يعني سيرفر كبير فيه قاعدة بيانات و مربوطه فيه اجهزة عن طريق الشبكة فيها البرنامج .
- في التير2 كانت الفكرة في عمل طبقتين منفصلتين الاولي هي الخادم و التي تتمثل في قاعدة البيانات و الثانية هي المخدوم و المتمثلة في البرامج التي تطلب الخدمة من قاعدة البيانات , ممكن ان تكون تلك الطبقتان علي اجهزة مختلفة او علي جهاز كمبيوتر واحد كطبقات من البرامج Software'sأما بالنسبة للأعمال المنطقية التي كان يقوم بها الخادم في التير 1 انتقلت من الخادم إلي المخدوم و بذلك تم حل مشكلة التحميل علي الخادم ولكن !!
- ظهرت مشكلة جديدة !!رغم أن هذا النظام عمل علي تخفيض التحميل علي السيرفر الا انه يطلب الكثير لقاء ذلك
  - أولا : برامج المخدوم لابد أن تكون موزعة وأيضا تدار على عدد كبير من المخدومين عبر المؤسسة ككل.
  - ثانيا : إذا حدث أى تغيير فى الأعمال المنطقية لأى سبب لابد من التغيير فى جميع المخدومين مما يؤدى الى استنزاف جهد و مال المؤسسة.أخيرا : نظام الخادم/المخدوم يكون سريع التأثر لتعطل الخدمات إذا حدث أي عطل من الخادم.



# تاريخ و هيكلية البرامج

**المرحلة الثالثة : Tier3-Application Server**

- جاء نظام الخادم/المخدوم ثلاثي الطبقات ليحل المشاكل الناتجة من النظام السابق فتتكون الطبقة الأولى من (Server) و الذي تحتوي على قواعد البيانات، والطبقة الثانية وهى الجديدة التى يوضع بها منطقية العمل (Business Logic) ثم بعد ذلك الطبقة الأخيرة وتكون للمستخدم (Client ) بحيث إن منطقية العمل وضعت فى طبقة مستقلة فلابد من وجود برمجيات وسيطة (Middleware ) وهى التى تربط الثلاث طبقات معا فبتالي يصبح client خالي من اي شي فقط هناك Browser .



# Service Oriented Architecture (SOA)
# معمارية الخدمية الموجهة

في أحيان كثيرة الحاجة ملحة للتعامل مع أكثر من نظام للحصول على خدمة ما على سبيل المثال نحتاج أن نرتبط بنظام آخر لكي نأخذ بعض البيانات و نستفيد منها ثم نمرر تلك البيانات إلى نظام آخر ففي قطاع الاتصالات مثلا: نقوم بتسديد الفواتير من خلال موقع المصرف نقوم بالدخول إلى نظام المصرف و من ثم المصرف يدخل إلى نظام سداد و سداد يتواصل مع شركة الاتصالات و شركة الاتصالات تقوم بتغيير في نظام الفواتير لديها و هكذا .

كل هذا يتم من خلال عملية واحدة طبعا لا نتوقع أن جميع هذه الانظمة تستخدم نفس التقنية فبعضها يعمل على الدوت نت و البعض الآخر على الجافا و غيرها يعمل على سي++ والسؤال الذي يطرح هنا كيف تتواصل كل هذه الانظمة مع اختلاف تقنياتها؟

فنجد ان كل نظام من هذه الانظمة يقدم بعض خدمات Service provider و يستفيد من أخرى service consumer فنظام سداد مثلا يقدم خدمة الربط إلى شركة الاتصالات و يقدمها إلى المصارف و يستفيد بدوره من خدمة تسديد الفواتير الموجودة في شركة الاتصالات فمن هذا المنطلق، هناك مكونات في البرنامج تقدم خدمات كما أن هناك مكونات أخرى تستفيد من هذه الخدمات و هذا ما يعرف بالـ SOA وهي معمارية الخدمة الموجهة .

# مثال معمارية الخدمية الموجهة

يعرض هذا المثال بنية تطبيق يستخدم خدمات الوِب في بناء تطبيق من تطبيقات الحكومة الإلكترونية. إن أهم فوائد تطبيقات الحكومة الإلكترونية هي تمكين المواطن من إنجاز الإجراءات إلكترونياً.

تتوفر مثلاً لدى هيئة الأحوال المدنية قاعدة بيانات تتضمن أسماء المواطنين وأرقامهم الوطنية ومعلوماتهم، ويتوفر لدى الإدارة العامة للجوازات قاعدة بيانات من نوع أوراكل Oracle، كما يتوفر لدى وزارة العمل قاعدة بيانات من نوع أكسس Microsoft Access. أما مصلحة الضرائب فليس لديها إلا ملف إكسل Excel بمعلومات الضرائب الخاصة بالمواطنين، ويُطلب ربط هذه المؤسسات ببعضها إلكترونياً لإتمام إجراءات المواطنين؛ فمثلاً عندما تحتاج مصلحة الضرائب إلى معلومات عن المواطن، يجب أن تستطيع الحصول عليها إلكترونياً من هيئة الأحوال المدنية.

قد يقترح المهندسون بناء قاعدة معطيات مركزية لهذا التطبيق، بحيث تُدمج جميع قواعد البيانات المؤسسات فيها، وتكون مشتركة لكي تستطيع جميع الجهات الوصول إليها. تُعدّ هذه الطريقة من الطرق الشائعة لبناء تطبيقات من هذا النوع، غير أنها تواجه عدة صعوبات تقنية منها:

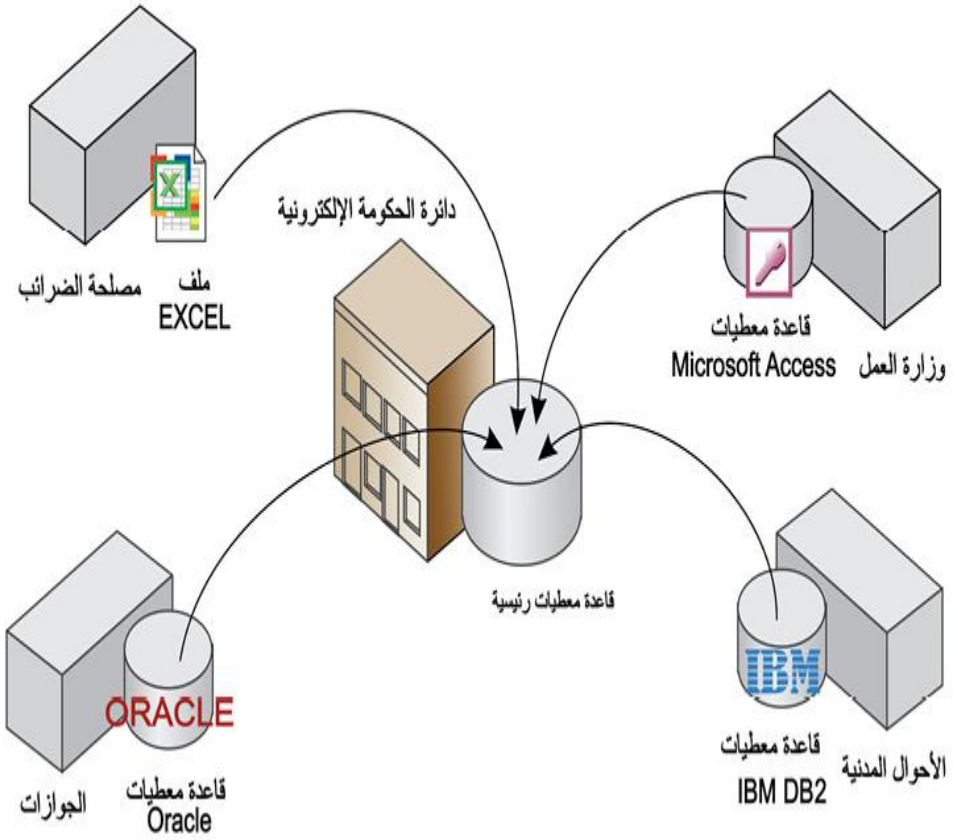
صعوبة دمج أنواع مختلفة من قواعد البيانات؛ إذ يحتاج الأمر إلى إعادة هندسة ووضع تصميم جديد لقاعدة البيانات.

وفي حال الدمج تصبح عملية تحديث قاعدة البيانات الجديدة بالمعلومات المتغيرة عملية معقدة. يبيّن الشكل (1) رسماً توضيحياً لبنيان التطبيق في هذه الطريقة، ويُعدّ بنياناً ذا قاعدة بيانات مركزية.

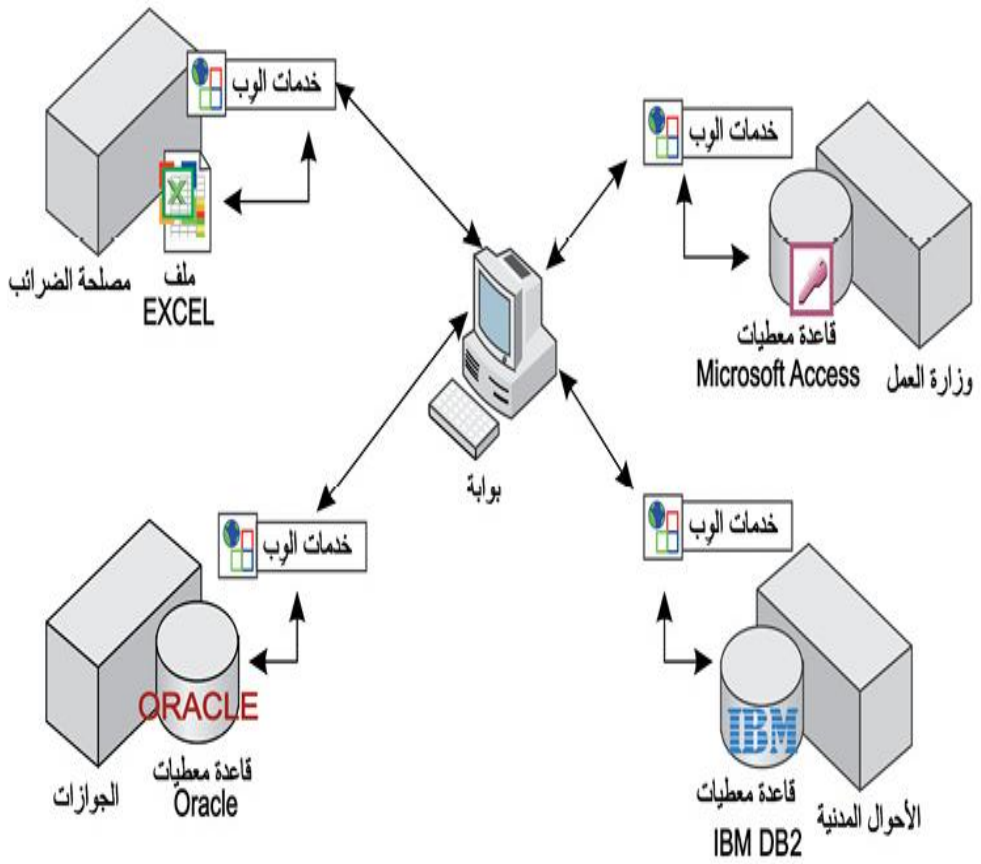
# مثال معمارية الخدمية الموجهة

لتفادي الصعوبات التقنية المشار إليها يمكن استخدام خدمات الوِب. تقوم كل مؤسسة ببرمجة خدمة الوِب الخاصة بها، ومن ثمّ يمكن للدوائر الاتصال فيما بينها عن طريق خدمات الوِب أو عن طريق دائرة مركزية تربط بينها وتضمن حماية الاتصال بين خدمات الويب، أو تُجمع هذه الخدمات في بوابة واحدة portal. يبيّن الشكل (2) رسماً توضيحياً لبنيان التطبيق في هذه الطريقة، وهو بنيان خدمي التوجه.

وفق الطريقة الأخيرة عندما تريد إدارة الجوازات مثلاً الاستعلام عن معلومات مواطن، ترسل طلباً لخدمة الوِب التابعة للأحوال المدنية. فتقوم هذه الخدمة بالرد بإرسال ملف Extensible Markup Language XML يتضمن المعلومات الخاصة بالمواطن، وذلك بغض النظر عن الطريقة التي ترتّب بها دائرة الأحوال المدنية معلومات المواطنين؛ إذ يكفي أن تعرِّف هذه الخدمة الواجهة البرمجية للاتصال بها والسماح بإرسال معلوماتها إلى الدوائر الأخرى أو إلى المستخدم مباشرةً. وبالمماثلة تعرّف مصلحة الضرائب واجهةً لخدمة الوِب لديها، وإن كانت المعلومات مخزّنة في ملف إكسل أو في ملفات نصية.




شكل (1)بنياناً ذا قاعدة بيانات مركزية




شكل (2)بنياناً خدمي التوجه

# Service Oriented Architecture SOA
# معمارية الخدمية الموجهة

فلو أخذنا مثال شركات الاتصالات أعلاه، يمكن ان تكون هناك العديد من الخدمات مثل خدمة الاستعلام عن فاتورة عميلGet Information Bill و خدمة الاستعلام عن بيانات العمل Get Customer Informationو خدمة تسديد الفاتورة مثلًا و خدمة تعديل بيانات عميل و خدمة طلب خط جوال و هكذا. كل هذه مكونات مستقلة بذاتها، أي انها تكون خدمة. و أهم شيء يجب ذكره هو أن كل خدمة تقوم بمهمة واحد فقط. و هناك شروط أخرى يجب توفرها مثل:

- أن تقوم كل خدمة بعمل مهمة واحدة فقط.
- أن تكون الخدمات Loosely coupled أي غير معتمدة على بعضها البعض و تغيير في الخدمة الأولى لا يؤثر أطلاقا على الخدمات الأخرى.
- Abstraction أي ان الخدمة تقوم بإخفاء طريقة العمل Logic الموجود بداخلها و لكن تأخذ مدخلات Input و تعطي نتيجة أو خدمة و هي Output
- Re-usability، أي ان الخدمة يمكن اعادة استخدامها في أكثر من مكان في البرنامج (أي لا تكون مصممة لخدمة برنامج واحد فقط).

مثال آخر على ذلك هو بناء نظام إدارة الموارد والذي يحوي الكثير من الأجزاء مثل المبيعات, الحسابات, المخازن, إدارة الموارد البشرية, التسويق والكثير غيرها فيمكن تصميم النظام بحيث يتم عمل كل برنامج بشكل مستقل وربطه ليكون الERP .

- بالتأكيد يمكن تجزئة النظام إلى مستوى أقل من ذلك بمعنى تجزئة المخازن مثلا ليحوي ثلاث تطبيقات مستقلة مثل المستخدمين وإدارة المخزون وإدارة الموظفين وربطهم ليكونوا نظام مخازن.
- هذا الأمر يعتمد على المصممين والمهندسين الذين يقومون بتصميم النظام ليحددوا المستوى المطلوب الذي سيتم به بناء النظام حيث أن التكلفة تزيد بطبيعة الحال كلما قمنا بتجزئة الأنظمة إلى خدمات ولكن هذه التكلفة تقل في إعادة إستخدام الخدمات في أماكن أخرى بدلا من تطويرها فمثلًا أن نقوم بإستخدام نظام المستخدمين في أكثر من تطبيق.

لذلك يتطلب تطبيق SOA وجود مصممين ومهندسين مهرة وكذلك وجود وقت كافي إلى حد تنفيذ التصميم فعليا.

- هناك مخططات مختلفة لتوصيف SOA وليس هناك نمط ثابت ولكن الفكرة العامة في القدرة على بناء خدمات مختلفة بشكل مستقل والتواصل فيما بينها لتمثل جسد واحد عند واجهة المستخدم.

# Service Oriented Architecture SOA
# معمارية الخدمية الموجهة

- ينقسم الى مبادئ اساسية و مبادئ تصميمية:-
- اولا المبادئ الاساسية و المتمثلة في:

1. Reuse: امكانية استخدام هذه الوحدات في عمليات مختلفة و ان تكون متوفرة عند الطلب.
2. Autonomous : عدم الاعتمادية عن استخدام هذه الوحدات على بعضها البعض عند تنفيذ العمليات .
3. Interoperability : القدرة على الاتصال ، وتنفيذ البرامج ، أو لنقل البيانات بين مختلف وحدات تقنية.
4. Composability: القدرة على تأليف و تركيب الخدمات.
5. Portability : القدرة على الانتقال من نظام إلى نظام أخر من دون أي تكاليف.
6. Standards compliance: الالتزام بالمعايير و المواصفات القياسية .
7. Services identification :قدرة على تحديد الخدمات.
8. categorization: القدرة على تصنيف و تبويب الخدمات.
9. delivery: القدرة على توصيل الخدمة إلى طالبيها.
10. monitoring and tracking :القدرة على المراقبة و المتابعة و التسجيل.

# المبادئ التصميمية للـ SOA

- Service encapsulation: و هي تلك المقدرة على تغليف مجموعة من الأوامر و التعليمات التي يمكن نقلها عبر الشبكات و الانترنت.
- Service loose coupling: عبارة عن الوصلات المحررة التي تقوم على المحافظة على العلاقات و تقليل الاعتمادية بينهما و تعتمد على الإدراك لمعرفة الخدمات المتوفرة.
- Service contract: وهو عقد الخدمة الذي يعرف طريقة الاتصال و يحتوى التعريفات اللازمة للخدمة.
- Service abstraction: هو تجريد الخدمة من أي معلومات تحوي كيفية عملها و تحتوي فقط على وصف الخدمة للعالم الخارجي .
- Service optimization: تقديم خدمات ذات جودة عالية.
- Service discoverability: قدرة الخدمة لكي يتم اكتشافها من قبل آليات الكشف عن الخدمات.
- Service autonomy : هي استقلالية الخدمة بحيث كل خدمة لديها القدرة على السيطرة على محتواها الخاص.

# أنماط المعمارية الموجهة نحو الخدمة

هناك ثلاثة أدوار في كل من اللبنات الأساسية للهندسة المعمارية الموجهة للخدمة:

1-مزود الخدمة

2-وسيط الخدمة

3-سجل الخدمة ، مستودع الخدمة ؛ وطالب الخدمة / المستهلك.

يعمل مزود الخدمة جنبًا إلى جنب مع سجل الخدمة ، ويناقش أسباب وكيفية الخدمات المقدمة ، مثل الأمان والتوافر وما يجب تحصيله وغير ذلك. يحدد هذا الدور أيضًا فئة الخدمة وما إذا كانت هناك حاجة إلى أي اتفاقيات تجارية.

يقوم وسيط الخدمة بتوفير المعلومات المتعلقة بالخدمة لمن يطلبها. يتم تحديد نطاق الوسيط من قبل من ينفذه.

يحدد طالب الخدمة الإدخالات في سجل الوسيط ثم يقوم بربطها بموفر الخدمة. قد يكونون قادرين أو لا يمكنهم الوصول إلى خدمات متعددة ؛ هذا يعتمد على قدرة طالب الخدمة.

## تنفيذ العمارة الموجهة نحو الخدمة:

يتم تنفيذ البنية الموجهة للخدمة (SOA )، عن طريق مجموعة واسعة من التقنيات التي يمكن استخدامها ، اعتمادًا على هدفك النهائي وما تحاول تحقيقه.

عادةً ما يتم تنفيذ البنية الموجهة للخدمة مع خدمات الويب ، مما يجعل "وحدات البناء الوظيفية يمكن الوصول إليها عبر بروتوكولات الإنترنت القياسية".

# المكونات الاساسية SOA

- **مستهلك الخدمة Service Consumer**

مستهلك الخدمة ممكن يكون مكون تطبيق , او برامج (Software) اخرى تطلب خدمة من تطبيقات مختلفة . مستهلك الخدمة يجد مقدم الخدمة في مكان يسمي الـ Service Registry او سجل الخدمة و يقوم بارسال طلب الخدمة و من ثم يتم تنفيذ وظيفة الخدمة.

- **مقدم الخدمة Service Provider**

هو عبارة عن تطبيق او برنامج (Software) موصول عبر شبكة يقوم باعطاء خدماته للمستهلكين ، و يستطيع المستهلك التعرف علي خدمته عندما يقوم مقدم الخدمة بنشر عقده ( Service Contract ) في سجل الخدمة لكي يتم التعرف عليه من قبل المستهلك.

- **عقد الخدمة Service Contract**

هي مواصفات من خلالها يصل مستهلك الخدمة الي مقدم الخدمة.

واجهة ٱلخدمة هَو ٱلعقد ٱلذي يَحدد هَوية وَقواعد طَلب ٱلخدمة. تَحتوي وَاجهة ٱلخدمة عَلى:

- بيانات رَسالة ٱلطلب
- بيانات رَسالة ٱلاستجابة
- شروط ٱلاستثناءات
- معلومات عَن وَظائف ٱلخدمة وَالغرض مَنه

- **Service Contract** يقوم بابلاغ مستهلك الخدمة عن الشكل المقبول لطلب الخدمة. Service Contract يخزن في Service Registry للسماح لمستهلكي الخدمات لرؤية و استخدام الخدمات المعروضة من Service Provider .

- **سجل الخدمة Service Registry**

هو عبارة عن سجل مربوط بشبكة اي يمكن الوصول اليه عن طريق الشبكة يقوم بقبول و تخزين ال Service Contracts من مقدميها و جعلها ظاهرة لمستهلكي الخدمات هذه .

# فوائد SOA

- تحسين العمليات التجارية .

من خلال جعل الخدمات متاحة في جميع أنحاء المؤسسة، يمكن لمجموعة من العمليات التجارية من الاستفادة من الوظيفة التي توفرها وحدات البرامج التي كان يتعذر الوصول إليها سابقا. باستخدام هذه الوحدات يصبح من الممكن مواصلة تحسين العمليات التجارية.

- خفض تكاليف تكامل النظام .

أصبحت ال Service Oriented Architectureمصممة لتربط التطبيقات المختلفة . فهي حلت مشكلة عدم تكامل المنظومة .

- امن المعلومات .

منذ ان اصبحت الخدمات تستخدم من قبل تطبيقات متعددة لها آليات الحماية الخاصة بها , اصبح هناك ما يضمن ان البيانات تمر عبر مراحل متعددة من التحقق علي صعيد الخدمة و المستخدم معا

- زيادة عائدات الاستثمار.

عبر تمكين الخدمات لمشاركة وظيفتها من الطبيعي ان تنعكس نتائج ذلك في زيادة عائدات الإستثمار من السفت ويرز . زيادة المرونة و الوظيفية للخدمات ستسمح للخدمات لكي تكون مستخدمة من قبل المؤسسة اكثر تلك التطبيقات التي عفا عليها الزمن .

# عيوب الـ SOA

- تزيد في تعقيد الانظمة.
- تزيد في الوقت لتنفيد.
- تحتاج إلى مجموعة أكبر من التراخيص .
- تحتاج إلى تعلم مجموعة كبيرة من التقنيات المتقدمة .

# ملخص

- كانت فكرة تطور البرامج في وضع نظام مستقل اي جزء فيه يقوم بعمل واحد , لكن في ظل الطفرة النوعية لتطور نظم المعلومات جاءت فكرة مستقلة عن الافكار التقليدية السابقة و هي كيف يمكننا صنع نظام متكامل متصل مع بعضه البعض يقوم علي اساس الاستخدامية اي يمكن ان نستخدم جميع و ظائف النظام الموجودة بدون الحاجة الي تصميمها من جديد .