ملخص العمل

تتصف خدمات الويب (Web Services) في عصرنا الحالي بالأهمية كونها تقدم خدمات عن بعد وتوفر الجهد والوقت على مستخدميها ، بالرغم من بعض جوانب القصور التي تتميز بها ووجود محاولات كثيرة لتحسين أدائها وتطوير آلية عملها .

ما يحصل اليوم في عصرنا دليل واضح على هذه المحاولا ت في اتجاه تطوير الويب بشكل عام وخدمات الويب (Web Services) بشكل خاص ، إذ أن فكرة الويب الدلالي (Semantic Web) ، تعد ثورة جديدة في عالم الانترنت اتجه نحوها الكثيرين لتطويرها والاستفادة من قدراتها في عدة مجالات ، وما خدمات الويب (Web Services) إلا واحدة من هذه المجالات التي تجري عليها محاولات للتغلب على محدوديتها وتطويرها في ذات الوقت .

تعتبر لغة توصيف خدمات الويب OWL-S إحدى نتائج الأفكار والجهود المبذولة في هذا الاتجاه وخطوة نحو بناء محركات بحث ذكية قادرة على فهم المحتوى وإدراك الحقائق و اكتشاف الخدمات وتركيب خدمات جديدة من خدمات موجودة موصفة دلالياً.

الفصل الأول

مقدمة

نتيجة لتزايد الطلب على خدمات الويب (Web Service) التي تلعب دوراً هاماً في تبادل المعلومات على شبكة الويب ، ونظراً لحاجة المجتمع باختلاف مجالاته لهذه التقنية التي توفر الوقت والجهد على مستخدميها ، زادت الحاجة إلى تطوير الويب لإيجاد وسائل أكثر تطوراً تساهم في سرعة البحث والوصول الدقيق إلى الخدمات المطلوبة ، فكانت ثورة الويب الدلالي (Semantic Web) ، التي من شانها أن تحل هذه المشكلة بقدراتها الدلالية ، وتحقق التوقعات المرجوة منها في الاكتشاف التلقائي للخدمات بفاعلية ودقة أكبر وتوفير آلية استدعاء دون التدخل البشري وتركيب خدمات جديدة تلبي طلب المستخدمين من خدمات بسيطة متوفرة .

في ورقة البحث هذه سنبحث في استخدام الويب الدلالي (Semantic Web) لتوصيف خدمات الويب دلالياً (Semantic Web Services) ، ففي :

الفصل الأول : مقدمه .

الفصل الثاني : سنشرح الويب الدلالي (Semantic Web) وتقنيات التي تستخدم لتمثيل نماذج البيانات ، كلغة وصف موارد الويب RDF ، ومخططات العلاقة مثل RDFS ، ولغة انطولوجيا الويب الـ OWL .

الفصل الثالث : سنشرح مجموعة أدوات الويب الدلالي (Semantic Web) التي تساعد على وصف مجال ما ، وتقديم نتائج منطقية يفهمها البشر ، كمحررات الانطولوجيا مثل Protégé ، والأطر البرمجية للويب الدلالي مثل Jena .

الفصل الرابع : سنشرح خدمات الويب ومقاييسها الأساسية .

الفصل الخامس : سنشرح لغة ترميز خدمات الويب OWL-S وعلاقتها مع لغة WDSL ، وآلية تطابق مع الـ UDDI .

الفصل السادس : الاقتراح بتطوير محرك بحث ذكي عن خدمات الويب (Web Services) الموصفة دلالياً ، وإمكانية بناء خدمات جديدة مركبة من خدمات موجودة متوفرة موصفة دلالياً .

الفصل الثاني

الويب الدلالي (Semantic Web)

مع الغنى الكبير الذي حققته شبكة الويب بما تحتويه من مصادر معلوماتية منوعة وإمكانية الوصول إليها عبر محركات البحث التقليدية ، إلا أن تنظيم هذه المعلومات والمستندات بصورة تسهل عملية البحث فيها والوصول إليها أمراً غاية في صعوبة بسبب محدوديات البحث وازدياد حجم المعلومات المنشورة على شبكة الويب .

من هذه المشكلة ظهرت فكرة الويب الدلالي (Semantic Web) أو الويب ذو معنى وهو القادر على وصف الأشياء بطريقة التي تمكن الحواسيب من فهمها . شيد الويب الدلالي (Semantic Web) من خلال ربط صفحات الويب الحالية إلى مجموعة البيانات المهيكلة والتي تشير إلى المعاني المرتبطة بهذه الصفحة .

يمكن للويب الدلالي (Semantic Web) أن يتألف من نماذج البيانات التي تستخدم عدد من التقنيات لتمثيلها منها :

- لغة تنسيق تبادل المعلومات مثل لغة وصف موارد الويب الـ RDF (Resource Description Framework) .
- مخططات العلاقة (Schema) مثل RDF Schema .
- لغة انطولوجيا الويب الـ OWL (Web Ontology Language) .

1. لغة إطار وصف الموارد RDF (Resource Description Framework)

تمثل لغة الـ RDF قاعدة أساسية لدعم الويب الدلالي ، تمَّ توصيتها من قبل هيئة المقاييس W3C (World Wide Web Consortium) عام 2004 ، وهي لغة بيانات وصفية بسيطة عمومية الاستخدام تبنى ضمن مستند XML ، لوصف المعلومات المحتواة في الويب .

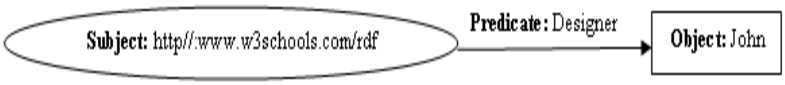
تستخدم لغة الـ RDF في وصفها للمصادر (Resources) الثلاثيات (Triples) :

- **Subject (Resource)** : وهو أي شيء يملك محدد URI (Uniform Resource Identifier) يمكن أن يوصف من قبل تعابير الـ RDF ، مثل صفحة الويب التالية : " http//:www.w3schools.com/rdf " .
- **Predicate (Property)** : هو مصدر (Resource) يملك اسم ويكون مستخدم كصفة (Property) ، وهذا الاسم يمكن أن يكون مستخدم لوصف بعض الجوانب المحددة أو المميزة للمصدر (Resource) . مثل كلمة (مؤلف Author أو مصمم Designer لصفحة ويب) .
- **Object (Property Value)** : وهو يمثل قيمة الـ Predicate ، كأن نذكر اسم شخص الذي صمم (Designer) صفحة الويب أو اسم مؤلف (Author) موقع الويب .

المثال التالي يظهر لنا أن صفحة الويب " http//:www.w3schools.com/rdf " ، قام بتصميمها شخص يدعى John :

```
<? xml  version = "1.0" ? >
< rdf : RDF        xmlns : rdf = " http://www.w3.org/1999/02/22-rdf-syntax-ns# "
                   xmlns : FR = " http://www.Exampel.com / rdfExampel# "
        < rdf : Description   rdf: about = " http//:www.w3schools.com/rdf ">
                   < FR : Designer > John < / FR : Designer >
        < / rdf : Description >
< / rdf : RDF >
```

يمكن تمثيلها بالشكل :



2. مخططات لغة وصف المصادر RDFS (RDF Schema)

على الرغم من إمكانيات لغة الـ RDF في وصف الموارد و تأكيد العلاقات بين تلك المصادر ، الأمر الذي يسمح للآلة بفهم المعلومات إلا أنها غير قادرة على تعريف الصفوف (Classes) وإدراك الصفوف الجزئية منها (subClasses) وتعريف الصفات الجزئية (subProperty) من الصفات (Property) المعرفة . هذه المحدودية لا تمنع اللغة من استخدام طريقة وصف تدركها الآلة أو تمنعها من قراءة البيانات واستنتاج القدرات لكنها لن ترقى إلى المستوى الذي نطمح إليه . إن جعل المعلومات الموزعة على الانترنت أكثر مرونة وقدرة آلية في المعالجة يتطلب الحاجة إلى مثل هذه المفردات . و ما نبحث عنه سنجده لدى لغة الـ RDFS (RDF Schema) التي تم توصيتها من قبل هيئة المقاييس W3C و التي يمكن النظر إليها كلغة تصف مفردات الـ RDF ، إن ما تفعله لغة RDFS هو التالي :

- إنشاء مفردات تصف الصفوف(Classes)، الصفوف الجزئية (SubClasses) ، الصفات (Properties) ، الصفات الجزئية (SupProperty) ... الخ.
- تربط الصفات (Properties) مع صفوفها (Classes) المعرفة إليها .
- تضيف دلالات إلى صفات (Properties) ومصادر (Resources) الـ RDF وتعرف المعنى لمصطلح موجود من خلال تحديد صفاته (Properties) وما هو نوع الكائنات (Objects) التي يمكن أن تكون قيم لتلك الصفات (Properties).

مفردات لغة الـ RDFS تكتب في مستند RDF ، ليس ذلك فقط بل تستخدم نفس نماذجها (الرسوم البيانية Graphs ، الثلاثيات Triples) ، بشكل آخر إن الـ RDFS هي امتداد للغة الـ RDF ، وباشتراك مفردات و تعابير اللغتين ستدفع الانترنت خطوة جديدة نحو الأمام باتجاه إمكانية القراءة والمعالجة الآلية .

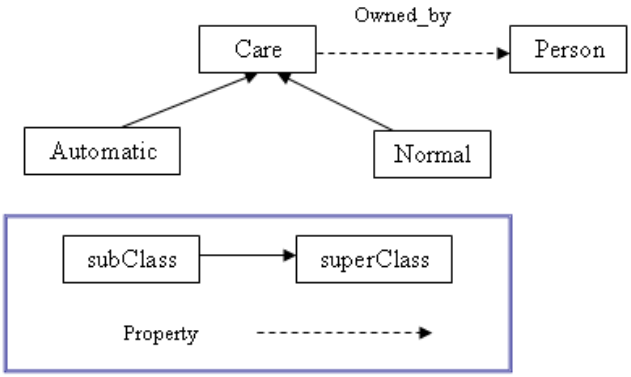
لنكتب الآن مثال نوضح فيه إمكانات اللغتين . تنشأ فكرة المثال على النقاط التالية :

- لدينا مصدر (Resource) يدعى **سيارة** (**Car**) ، و آخرين يدعى الأول **عادي** (**Normal**) والثاني يدعى **اتوماتيك** (**Automatic**) ، وكلاهما مصادر جزئية من المصدر (Resource) **سيارة** (**Car**) .

- المصدر (Resource) سيارة (Car) يملك صفة تدعى : **ممتلكة من قبل (Owned_by)** والتي تأخذ قيمها من المصدر **شخص (Person)** .

وفي ما يلي كود المثال :

```
<?xml version="1.0"?>
        <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                 xml:base="http://www.exampelCare.com /WorledCar /Care.rdfs">
// classes definitions
        <rdfs:Class rdf:ID="Car">
        </rdfs:Class>
        <rdfs:Class rdf:ID="Person">
        </rdfs:Class>
        <rdfs:Class rdf:ID="Automatic">
                <rdfs:subClassOf rdf:resource="#Care"/>
        </rdfs:Class>
        <rdfs:Class rdf:ID="Normal">
                <rdfs:subClassOf rdf:resource="#Car"/>
        </rdfs:Class>
// property definitions
        <rdf:Property rdf:ID="owned_by">
                <rdfs:domain rdf:resource="#Car"/>
                <rdfs:range rdf:resource="#Person"/>
        </rdf:Property>
        </rdf:RDF>
```



من خلال هذا المثال تظهر لنا إمكانيات لغة الـ RDFS في تعريف الصفوف الجزئية (subClasses) من الصفوف (Classes) المعرفة ، كما تمكنًا من تحديد منطلق (Domain) الصفة Owned_by والقيم التي ستأخذها (المستقر Range) .
كما يجب أن نعلم ، في الانطولوجيا صفات الآباء ترثها الأبناء و بالتالي فإن صفة (Owned_by) سترثها كل من الصف (Normal) والصف (Automatic) .

عند الالتزام بترميز مجال ما بثلاثيات (Triples) لغة الـRDF ومفردات لغة الـ RDFS ، لن يصبح من الصعب على الآلة إدراك المعنى وفهم المعلومات الموزعة على الويب لتكون عامل بناء للمساعدة على توصيف مجال ما بصفوفه وصفاته وعلاقاته وهذا ما يسمى بالانطولوجيا (Ontology) .

للانطولوجيا (Ontology) الكثير من العبارات التي تعرفها ، ولكل منها وجهة نظر من منظور مختلف في عالم الويب الدلالي (Semantic Web) .
تعرف الانطولوجيا (Ontology) أو ما يعرف بعلم دراسة المعاني من قبل هيئة المقاييس العالمية W3C :
هي توصيف رسمي لمجال معين من خلال تحديد المفاهيم (Concept) الخاصة به وصفاته (Properties) والقيود المطبقة (Structures) عليها والعلاقة بين هذه المفاهيم .
تكون المفاهيم (Concepts) عادة منظمة بشكل هرمي بحسب علاقة التعميم والتخصيص وفي هذه الهرمية يكون المفهوم الخاص ابناً للمفهوم الأعم ويملك كامل الصفات وميزات المفهوم الأعم ، حيث أن هذه الصفات تصف الميزات والخصائص المختلفة للمفاهيم كما تستخدم لربط المفاهيم المختلفة معاً .

3. لغة انطولوجيا الويب (OWL) Web Ontology Language

على رغم من أن لغة الـ RDFS مؤثرة بشكل جيد في الانطولوجيا إلا أنها لا تزال تحوي الكثير من المحدوديات ، فعلى سبيل المثال إذا وجد مفهومان (Concepts) مختلفان بالتسمية يتكلمان عن نفس الموضوع في مستندين مختلفين فإن قوة المحاكمة المنطقية تكون أقوى لو استطعنا الإشارة إلى أن هذين المفهومين متكافئين.

مثال آخر ، قد نحتاج أحياناً إلى التعبير عن الانفصال بين المفاهيم لتأكيد أنهما شيئين منفصلين تماماً ، وهذا مهم جداً لنؤكد وعلى سبيل المثال أن كل من الجنسين (ذكر Male ، أنثى Female) هما مفهومان منفصلان تماماً .

إن ما ذكرناه على سبيل المثال لا الحصر ، لا يمكن تمثيله من خلال لغة الـ RDF و RDFS ، لذلك كانت هناك الحاجة إلى تمديد مستنداتها . بشكل أدق من ذلك نحن نحتاج إلى لغة أكثر تقدماً لتغطي فجوات ما سبقها من لغات ، هذه اللغة الجديدة تدعى لغة انطولوجيا الويب OWL (Web Ontology Language) ، تم توصيتها من قبل هيئة المقاييس العالمية W3C ، وربما تكون اللغة الأكثر شهرة وانتشاراً ، وهي مبنية أساساً على مستندات لغة الـ RDFS ، بمعنى أنه يمكن تمثيل لغة الـ OWL بالشكل :

OWL = RDFS + New Constructs For Expressiveness

وبالتالي فإن جميع المفاهيم (Concepts) والصفات (Properties) المتوفرة من خلال مستندات الـ RDFS يمكن استخدامها عند إنشاء مستندات الـ OWL .

تعتمد هذه اللغة على قاعدتين أساسيتين هما :

1. **فرضية العالم المفتوح (Open world assumption)** : أي أنه لا يمكن الافتراض أننا نعرف كل المعلومات عن مجال معين وبالتالي عدم توفر معلومة معينة لا يعني نفيها .
2. **فرضية لا أسماء فريدة (NO Unique Name assumption)** : لا نستطيع القول أن المصادر مختلفة بسبب اختلاف المحددات URI ، ما لم يذكر خلاف ذلك صراحة .

لنأخذ المثال التالي نوضح فيه قوة الـ OWL ، تتمثل فكرة المثال بالنقاط التالية :

- لدينا المفهوم (Concept) شخص (Person) لديه صفة (يملك اسم hasName) ، ومفهومان جزئيان منه هما :

طالب (Student) ، مدرس (Professor)

سنعتبر أن المفهوم (Concept) طالب (Student) منفصل عن المفهوم (Concept) مدرس (Professor) وهذا ما سيعبر عنه من خلال **owl: DisjointWith** .

- لدينا المفهوم (Concept) مقرر (Module) .
- لدينا الصفة (يعلم teach) التي منطلقها (**owl: Domain**) مدرس (Professor) و مستقرها (**owl: Range**) مقرر (Module) ، أي تحدد لكل مدرس المقررات التي يدرسها ، وهناك صفة أخرى تدعى الخاصية (تدرس من قبل taughtBy) تتناسب عكساً مع صفة (يعلم teach) وهذا ما نعبر عنه من خلال **owl: InverseOf** .
- كما سنؤكد من خلال المثال أن جميع القيم التي ترتبط بالمفهوم (Concept) مدرس (Professor) يجب أن تكون من المفهوم (Concept) مقرر (Module) من خلال صفة (يعلم teach) وهذا ما نعبر عنه عبر owl: allValueFrom.

وفي ما يلي كود المثال :

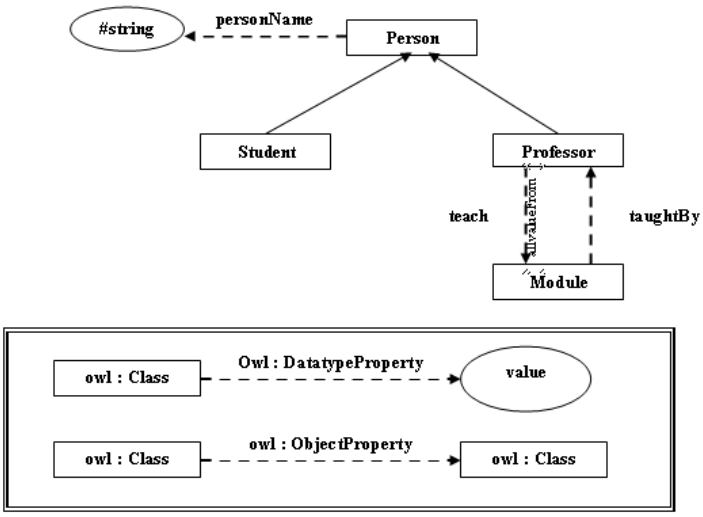
```
<?xml version = "1.0"?>
< rdf : RDF
        xmlns :rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns :my="http:// www.something.com / myontology #"
        xmlns :owl="http://www.w3.org/2002/07/owl #"
        xmlns : rdfs="http://www.w3.org/2000/01/rdf-schema #">

        <owl : Ontology rd f: about="http:// www.something.com / myontology.owl"/>

// Classes Definitions

        <owl : Class rdf : ID=" Person " >
        </owl : Class >
        <owl : Class rdf : ID ="#Module"/>
        <owl : Class rdf : ID=" #Student">
                        <rdfs : subClassOf  rdf : resource=" #Person"/>
                        <owl : DisjointWith rdf : resource = " #Professor "/>
        </owl : Class>

        <owl : Class rdf : ID ="#Professor">
                <rdfs : subClassOf  rdf : resource="#Person"/>
```

```
                <rdfs : subClassOf>
                        <owl : Restriction>
                                <owl : onProperty rdf : resource="# teach "/>
                                <owl: allValuesFrom rdf : resource="# Module"/>
                        </owl : Restriction>
                </rdfs : subClassOf>
        </owl : Class>
```

// Properties Definitions

```
        <owl : ObjectProperty rdf : ID="#teach">
                <rdfs : domain rdf : resource = "#Professor"/>
                <rdfs : range rdf : resource="#Module"/>
                <owl : inverseOf>
                        <owl : ObjectProperty  rdf : ID = "# taughtBy "/>
                </owl : inverseOf>
        </owl : ObjectProperty>
        <owl : DatatypeProperty rdf : ID =" personName ">
                <rdfs : domain rdf : resource="#Person"/>
                <rdfs : range rdf :
        resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl : DatatypeProperty>
</ rdf : RDF>
```

الشكل البياني :



الفصل الثالث

1. محررات الانطولوجيا (Ontology Editors)

بعد أن قمنا بكتابة مستندات الانطولوجيا ، أصبح الآن من الممكن قراءة المعلومات منها واستنتاج القدرات الدلالية وتنفيذ المحاكمات المنطقية عليها ، بمعنى أصبح هناك حاجة إلى وجود أدوات قادرة على تحرير الانتولوجيا (Editors) وتنفيذ المحاكمات المنطقية (Reasoning Engine) على تلك المستندات . الأمر الأكثر مثالية إلى مثل هذه الأدوات هو إنشاء واجهة تطبيق برمجية API (Application Programming Interface) ، قادرة على تصور وتمثيل بنية لغات الانطولوجيا (RDF , RDFS , OWL) وتوفير الخدمات التي قد نحتاجها . هذه الأدوات تمثل جزء من بيئة تطوير الويب الدلالي .

الأمثلة التي ذكرناها جميعها كان يتم تحريرها في ملف نصي كالمفكرة (Notepad) ، لكن في الحقيقة هناك الكثير من أدوات تحرير الانطولوجيا (Editor) التي تجعل من إنشاء مستندات الانطولوجيا أمراً سهلاً . تلك المحررات تشبه إلى حد كبير بيئات تطوير متكاملة IDEs (Integrated Development Environments) المستخدمة في لغات البرمجة كلغة الـ Java مثل (J Builder) التي تسمح لنا بإنشاء كود برمجي وتكوين واجهة تطبيق و استيراد ملفات من تنسيقات مختلفة كملفات XML ...

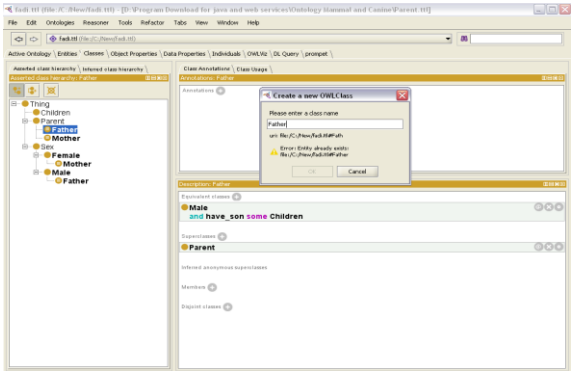
وعلى هذا النحو تقوم محررات الانطولوجيا بتقديم تصور وتمثيل للصفوف (Classes) والصفات (Properties) والأفراد (Instances) وإنشاء العلاقات والقيود عليها ، مع إمكانية تنفيذ محاكمة منطقية من مستوى معين ، واستيراد مستندات انطولوجيا ، ورسم الشكل البياني لانطولوجيا معينة بالإضافة إلى خدمات أخرى .

من أشهر هذه المحررات أداة مفتوحة المصدر من جامعة ستانفورد – كاليفورنيا تدعى Protégé ، والتي تسمح لنا ببناء انطولوجيا من خلال إنشاء المفاهيم (Concepts) والعلاقات الهرمية لها والخصائص (Properties) و الأفراد (Individuals) والقيود (Structures) المطبقة عليها... ضمن مجموعات تبويب (Tab) خاصة لكل منها .

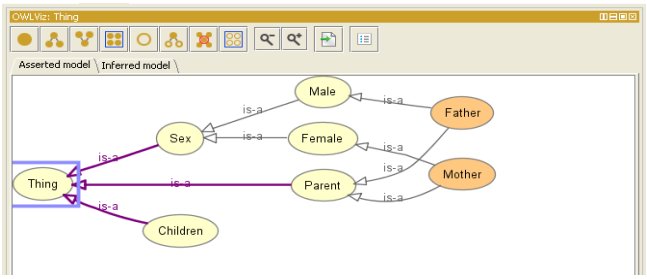
كما تسمح لنا الـ Protégé باستيراد مستندات الانطولوجيا و استعراضها وبناء الشكل

البياني لها وتنفيذ محاكمات منطقية على مستوى معين من المحاكمة بالإضافة إلى خيارات أخرى .

الشكل التالي يظهر لنا واجهة أداة الـ Protégé :



الشكل البياني لانطولوجيا منشأة :



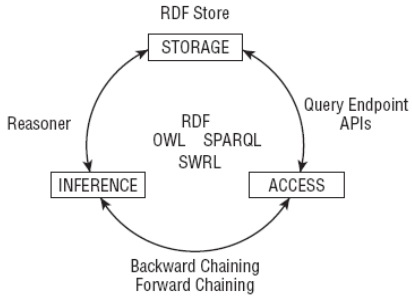
بعد تكوين الانطولوجيا ، نقوم بحفظها في ملف يمكن تحريره من خلال محرر النصوص (Notepad) لمشاهدة كود الانطولوجيا (Ontology) المنشأة .

2. الأطر البرمجية للويب الدلالي(Framework Semantic Web)

قد يتبادر إلى ذهن القارئ ، ما هو الغرض من الانطولوجيا (Ontology) وهل هي تطبيق (Application) قادرين على تنفيذه ؟! في الحقيقة وكما ذكرنا سابقاً ، الانطولوجيا (Ontology) تصف مجال ما مستخدمة الدلالات الواضحة ، ومن هنا نجد أنها مجرد لغة لوصف الأشياء وليست تطبيق يمكن استخدامه كما في التطبيقات البرمجية الحالية ، هذا يقودنا للقول أن الانطولوجيا (Ontology) وحدها لا تفعل شيء . وبالتالي لابد من وجود إطار برمجي للويب الدلالي (Semantic Web Framework) يسمح للتطبيقات من استثمار قدرات الانطولوجيا الوصفية في تعريف الدلالات والمعاني .

هذه الأطر البرمجية تتألف من مجموعة مكونات متكاملة فيما بينها تسمح بإنشاء واستعمال قواعد المعرفة (Knowledgebase) وتوفر إمكانية ، تخزين (Storage) ، استعادة (Retrieval) معلومات الـ RDF ، بالإضافة إلى ترجمة دلالات الانطولوجيا (Ontology) .

تتركب عادة هذه الإطارات من ثلاثة مكونات (Components) رئيسية وهي : التخزين (Storage) ، الوصول (Access) ، الاستدلال (Inference) أو نظم المحاكمة (Reasoner) والشكل التالي يوضح تفاعل هذه المكونات مع بعضها :



قواعد المعرفة (Knowledgebase): هي مجموعة حقائق (Facts) و هي نوعان :

- حقائق واضحة (Facts Explicit) : هي البيانات (Statement) المؤكدة .
- حقائق مضمنة (Facts Implicit) : يشار إليها ضمنياً و المشتقة أصلاً من خلال المحاكمة (Reasoning) .

التطبيقات التي تنجز الاستدلال (Inference) غالباً ما يشار إليها بنظم المحاكمة (Reasoning Engines أو Reasoner) وهي النظام الذي يستخلص الحقائق (Facts) الضمنية انطلاقاً من الحقائق الواضحة المؤكدة في قواعد المعرفة (Knowledgebase) وفقاً لقواعد محددة نذكر منها :

Rules and Rule Engine , RDF Store , Decision Tree …

يوجد العديد من إطارات الويب الدلالي المتوفرة اليوم ، من أهمها وأكثرها شيوعاُ هي **Jena** طورتها مختبرات الـ **hp** وهي عبارة عن إطار برمجي لبناء تطبيقات الويب الدلالي ، توفر بيئة تطوير برمجية لكل من RDF , RDFS ,OWL بما في ذلك أسس قواعد نظم المحاكمة (Reasoning Engines) . تستخدم Jena بشكل واسع مع العديد من المشاريع وخاصة تطبيقات الـJava لبساطتها.

كتبت Jena بلغة Java وهي تعتمد عليها لترجمة عبارات (Statement) وتراكيب (Construct) الويب الدلالي إلى نحو أعمال برمجية مفيدة كبناء صفوف (Classes) والكائنات (Object) واستدعاء الطرق (Method) والسمات (Attributes) .

تمتلك Jena صفوف (Classes) Java الرئيسية التالية :

Resource,Statement,Graph ,Model,Query and ResultSet ,Reasoner

بالإضافة إلى صفوف لتراكيب خاصة من الويب الدلالي (Semantic Web) .

سنوضح على سبيل المثال آلية استخدام نظم المحاكمة :

من السمات المهمة والمطبقة على بيانات الويب الدلالي (Semantic Web) هي تطبيق نظم المحاكمة (Reasoner) عليها لإدراك كامل قدراتها الدلالية ، من أجل ذلك نحن نحتاج إلى تطبيقها على ما هو موجود من بيانات فعلية في الـ Model (وهو الخطوة الأولى لمعالجة بيانات الويب الدلالي (Semantic Web) يستخدم ليشير إلى مكانها ، فبدون تحديد المكان لا يمكن تطبيق الأفعال عليها كالاستعلام) .

يقدم الإطار البرمجي لـ Jena ثلاثة أنواع من نظم المحاكمة (Reasoner) :

1. نظم محاكمة داخلية مبنية ضمن الإطار البرمجي لـ Jena .
2. نظم محاكمة خارجية تعرض كملفات Java خارجية مثل الـ Pallet .
3. نظم محاكمة خارجية عن بعد تعرض من خلال هيكل DIG . (DIG DL Information Group هو هيكل XML موحد لوصف نظم المحاكمة المنطقية عبر بروتوكول الـ http) .

الكود التالي يوضح لنا طريقة استخدام نظم المحاكمة الداخلية لإطار Jena :

```
Private InfModel bindJenaInferenceReasoner ( )
{
Reasoner reasoner = ReasonerRegistry.getOWLReasoner ( ) ;
reasoner = reasoner.bindSchema ( modelMem ) ;
InfModel inferredFriends = ModelFactory.createInfModel ( reasoner , modelDB ) ;
return inferredFriends;
}
```

تقدم لنا طريقة الاستدعاء ( ) ReasonerRegistry.get محاكمة منطقية من نوع OWL المحتواة في إطار Jean ، بعد ذلك قمنا بربط نظم المحاكمة (Reasoner) إلى الـ Model المعطى والذي يحمل الـ Ontology أو الـ Schema .

ثم قمنا بعد ذلك باستدعاء الطريقة (P1 , P2)ModelFactory.creatInfModel لإنشاء نموذج جديد الذي سندمج فيه المحاكمة المنطقية (Reasoner) مع الـ Model والموجودين كوسيطين ، وعند دمجهما فإن النموذج الناتج سيحتوي على اثنان من كائنات الـ Graph .

الكائن الأول يحتوي نموذج البيانات الأصلية ، والكائن الثاني يحتوي فقط على البيانات المشتقة (الناتجة من المحاكمة) .

إن هذا المثال يوفر لنا البيانات المشتقة التي تغطيها تراكيب الـ OWL مثل :

sameAs , SymmetricProperty , maxCardinality …

الاستدلال غير واضح و يحتاج لتوضيح باستخدام مثال

علاقة الأمومة = صف جزئي من علاقة الأبوة

سمر أم هاني

من هم آباء هاني؟ = سمر

بالرغم من عدم وجود علاقة أبوة بين سمر و هاني

لكن يوجد علاقة أمومة و هي جزء من علاقة الأبوة

الفصل الرابع

1. خدمات الويب Web Service

هندسة بناء خادم – عميل (Client - Server) منذ فترة طويلة تعتبر الخيار الأفضل لبناء تطبيقات موصفة ، وما خدمات الويب إلا واحدة من هذه الهندسة ، فهي نظام برمجي مصمم لدعم قابلية تفاعل حاسوب إلى حاسوب عبر شبكة الانترنت لتبادل المعلومات . إذا خدمة الويب يجب أن تفعل شيء كجزء من هذا التفاعل كأن تقدم لنا معلومات أو تحول بيانات ، بكلمة أخرى الخدمات وجدت لتوفر لنا خدمة . يعتمد معظم التفاعل المحيط بخدمات الويب بالحصول على النتيجة مع اختلاف بيئات العمل:

- ليس هناك أهمية لنوع الحاسوب أو نظام التشغيل .
- ليس هناك أهمية لنوع اللغة التي كتب بها التطبيق .

من هنا نجد الأهمية الحقيقة وراء خدمات الويب وانتشارها الواسع في عالم الويب ، بالرغم من ما تواجهه من مشاكل وتحديات . إحداها معالجة الاتصال بين خادم –عميل (Client - Server) وقد كان الحل هو استخدام بروتوكول نقل نصوص التشعبية HTTP (HyperText Transfer Protocol) لسببين :

- HTTP موجود في كل مكان .
- HTTP قادر على العبور من خلال الجدران النارية (Firewalls) .

والمعايير الأساسية لخدمات الويب بنيت أساساً من هذا الحل ، وعن طريق الالتزام بهذه المعايير يمكن لخدمات الويب أن تحقق التفاعل عبر الشبكة . هذه المعايير هي :

1. لغة وصف خدمات الويب WSDL (Web Services Description Language) .
2. بروتوكول الوصول البسيط إلى الكائنات SOAP (Simple Object Access protocol) .
3. التكامل والوصف والاكتشاف العام UDDI (Universal Discovery Description Integration) .

1.1. لغة وصف خدمات الويب WSDL (Web Services Description Language)

هي لغة تعتمد أساساً على لغة الـ XML لوصف الخدمة (اسم الخدمة ، وظائف الخدمة ، بروتوكول النقل ...) ، بكلمة أخرى هو مستند يعلن عن الخدمة التي يصفها ، يتم الوصف بصورة نظرية مستقلة عن لغة البرمجة التي يكتب بها التطبيق وعن نوع الحاسوب ونظام التشغيل الذي يتم العمل عليه ، وهذا سبب اعتمادها في الوصف على لغة الـ XML . في أغلب الأحوال مستندات الـ WSDL تولد تلقائياً عبر الخادم المضيف (Host Server) لخدمة الويب ، وهو ينقسم منطقياً إلى مجموعتين :

- الوصف المجرد (Abstract) : يتم فيه وصف إمكانيات خدمة الويب (الدخل ، الخرج ، العمليات ...) ، وهو يتكون من أربع عناصر XML وهي :
  Types , Message , Operation , PortType
- الوصف الملموس (Concrete) : يتم فيها تخزين العديد من التفاصيل الخاصة بكيفية إنشاء اتصال فعلي بالخدمة (البروتوكول، تنسيق الرسائل المتبادلة ، النقل ، المعالجة) ، هناك ثلاثة عناصر XML تمثلها وهي :
  Service , Port , Binding

هناك أيضاً العنصر الجذر < Definitions > والذي يحوي كل شيء يجب أن يعرف في المستند ، بالإضافة إلى هذه العناصر سنرى رسائل الـ SOAP وتعريفات مخطط بيانات الـ XML (XSD) في مستند الـ WSDL .

كون مستند الـ WSDL يعتبر بمثابة الإعلان عن الخدمة ، فلابد أن يكون مورد هام للمعلومات التي يحتاجها طالب الخدمة لإدراك فيما إذا كانت هي الخدمة المطلوبة .

لكن بالنظر إلى المستند ، لا نجد الكثير من الدلالات التي تحقق مثل هذا الهدف ، وقد تمثل معلومات الدخل (Input) والخرج (Output) إحداها إن لم تكن الوحيدة ، لكنها مع ذلك قد تتشابه لدى الكثير من الخدمات وبالتالي ستصبح عديمة الأهمية عند البحث عن خدمة معينة.

1.2. بروتوكول الوصول البسيط إلى الكائنات SOAP (Simple Object Access protocol)

هو بروتوكول قياسي يحدد كيف ينبغي لتطبيق أن يتصل ويتبادل المعلومات مع تطبيق آخر على الانترنت ، إن أهمية الـ SOAP تكمن في علاقتها مع الـ HTTP .

فالـ HTTP مصمم فقط من أجل الاستخدام بين متصفحات الويب (Web Browsers) وخادم الويب (Web Server) ، بمعنى هو الاتصال بين الحواسيب على الانترنت التي تحدث بشكل رئيسي على متصفحات الويب (Web Browsers) ، لكن عندما نريد تبادل المعلومات بين الحواسيب والتي هي آلية اتصال أكثر تعقيداً و أعلى مستوى ، فإننا بحاجة إلى بروتوكول اتصال يؤمن لنا تبادل المعلومات بحرية أكثر ، لهذا الهدف اخترع بروتوكول الـ SOAP .

فإذا كان تطبيق على حاسوب ما أرسل بيانات من خلال بروتوكول الـ SOAP فإن تطبيق آخر على حاسوب آخر سيكون قادر على الاستفادة من البيانات دون مشكلة ، بغض النظر عن لغة البرمجة المستخدمة في كل واحد من هذه التطبيقات أو أنظمة التشغيل التي تعمل عليها هذه التطبيقات ، كون رسائل الـ SOAP بحد ذاتها هي رسائل XML .

1.3. التكامل والوصف والاكتشاف العام UDDI (Universal Discovery Description Integration)

قد نرغب أحيانا في الحصول على خدمة محددة للوصول إلى هدف معين ، ولكن المشكلة تكمن أحياناً في أننا لا نعرف فيما إذا كانت الخدمة موجودة أو غير موجودة ، الأمر الذي يدفعنا إلى البحث عنها ، وهي الخطوة الأولى في اكتشاف الخدمة . يقدم لنا المكون الثالث من المقاييس الأساسية لخدمة الويب حلاً مثالياً لهذه المشكلة ، فهو مصمم خصيصاً لاكتشاف خدمات الويب ، وهو يدعى بمقياس التكامل والاكتشاف والوصف العام الـ UDDI (Universal Discovery Description Integration) ، وظيفته تقديم الدعم لنشر وإيجاد الخدمات الموصفة ، وهو يملك تركيب يمكن المختصين من تسجيل الخدمات والبحث عنها. لهذا الدليل أربع تراكيب بيانات رئيسية:

- businessEntity : بيانات هذا التركيب تستخدم لتمثيل معلومات مقدم الخدمة ، يملك هذا التركيب خاصية تدعى businessKey تستخدم لتحديد معرف وحيد لمقدم الخدمة يولد تلقائياً . يدعى هذا الـ Key بـ UUID (Universally Unique Identifier) ، وهناك أدوات لتوليد هذه الأرقام .
- businessService : وهو يمثل عادة خدمة معينة مقدمة من قبل تركيب businessEntity ، إذاً هو المدخل الحقيقي إلى الخدمة التي نهتم بها . يملك خاصية serviceKey تستخدم لتحديد معرف وحيد لهذه الخدمة وهو يولد تلقائياً.
- bindingTemplate : يعرف المعلومات التقنية كالـ Interface للخدمة و محدد URL للـ end Point ... أيضاً يملك هذا التركيب خاصية تدعى bindingKey تستخدم لتحديد معرف وحيد لهذا التركيب وهو يولد تلقائياً.
- tModel : هو أول صف في التراكيب المذكورة سابقاً ، مما يعني أنه لا يحتوي تركيب ولكن يمكن رؤيته في أي تركيب آخر في الـ UDDI . كل tModel يملك خاصية تدعى tModelKey لتحديد معرف وحيد لها يولد تلقائياً .

الـ tModel يمثل نوع الخدمة (Service Type) والتي هي ببساطة الـ Interface المسجلة ضمن مستودع الـ UDDI .

عندما نريد البحث عن خدمة معينة غير مدركين لوجودها أو عدم وجودها بالإضافة إلى عدم امتلاكنا لمعلومات قد تصلنا بها مثل (اسم الخدمة ، اسم مستند الـ WSDL أو موقعه , tModelKey , businessKey , serviceKey , bindingKey) ، فإنه يصبح من الصعب جداً اكتشاف الخدمة . لقد أدرك مصممي الـ UDDI هذه المشكلة ، لذلك قاموا بتحسين قدرة البحث من خلال تضمين مسجل الـ UDDI نوعين من المعلومات الإضافية وهي معلومات لتصنيف الخدمة والتعريف بها من خلال عنصرين مضمنان في التراكيب : identifireBag , categoryBag .

يوفر لنا الـ UDDI مكاناً مثالياً لاحتواء خدمات الويب ، لكن استخدام الـ UDDI في اكتشاف الخدمة المطلوبة ،أمر بالغ الصعوبة إذا لم يكن مستحيلاً ، للأسباب التالية:

- البحث في الـ UDDI يستند على الكلمات المفتاحية ، بمعنى أما أن تجد العديد من الخدمات أو لا تجد شيء .
- تنوع مخططات التصنيف والتعريف ، يتطلب قدر معين من الإدراك من قبل الباحثين لها ، لكن يبقى احتمال حدوث اختلاف في التصنيف من قبل مجموعة مقدمي الخدمات الأمر الذي سيجعل من تواجد المخططات أمر فاشل تماماً .
- الـ UDDI لا يقدم أي طريقة بحث عن خدمات الويب .

صعوبة العثور على الخدمة المطلوبة حتماً لا يزال بوصفه واحد من الدوافع الرئيسية لرؤية الويب الدلالي ، فصعوبة البحث عن المعلومات هي التي جعلت الناس تبدأ تفكر حول إضافة الدلالات إلى صفحات الويب لجعل البحث أكثر جدوى وكفاءة ، وهذا يفتح توقعات جديدة في عالم خدمات الويب :

- الاكتشاف التلقائي لخدمات الويب (Discovery) : لجعل الوصول إلى خدمة ما أمراً ناجحاً فإنه ينبغي توفير طريقة اكتشاف واضحة وأن تكون طريقة تلقائية بدرجة كبيرة من الدقة والكفاءة العالية .
- الاستدعاء التلقائي لخدمة الويب (Invocation) : بعد اكتشاف الخدمة لابد أن يكون هناك قدرة على استدعاء الخدمة بشكل تلقائي ، فبعض التطبيقات

تعمل بشكل مستمر دون انقطاع وعملها يتعلق باستمرار استدعاء خدمة معينة في فترات زمنية معينة .

- التكوين التلقائي لخدمات الويب (Composition) : في بعض الأحيان عمل معين يحتاج إلى طلب عدة خدمات ويب للعمل معاً ، وبالتالي فإنه لابد أن يكون هناك آلية واضحة لإيجاد جميع الخدمات الضرورية واستدعائها بالترتيب الصحيح لتحقيق الهدف من هذا العمل .
- المراقبة التلقائية لعمليات التنفيذ (Monitoring) : إذا كان جميع ما سبق من العمليات تتم تلقائياً ، عندئذ كيف لنا أن نعلم فيما إذا كانت الخدمة المطلوبة قد نفذت بنجاح ، لذلك لابد من أن يكون هناك آلية للكشف و الإبلاغ عن الفشل المحتمل إن وجد .

ما سبق ذكره جعل الباحثين والمطورين يبدءون بالنظر فيما إذا كانت بعض الدلالات والمعاني في عالم خدمات الويب (Web Services) يمكن أن تساعدهم في هذا المضمار . فكانت أولى الخطوات النظر إلى المقاييس الأساسية لرؤية فيما إذا كان هناك معلومات دلالية حاضرة فيها .

- **SOAP** : هو مواصفة من أجل تأمين تبادل البيانات والاتصال على شبكة الانترنت ، وهو يعمل بعد اكتشاف الخدمة ، لذلك سوف لن يساعدنا كثيراً .
- **WSDL** : تصف خدمة ويب وهو يمثل المكان المثالي للتنقيب عن المعلومات ، ولكن في الحقيقة ليس هناك دلالات حقيقة في مستندWSDL تساعدنا في البحث.
- **UDDI** : مسجل الخدمات ، وهو كما في الـ WSDL لن نجد فيه الدلالات التي تقدم لنا الفائدة الحقيقية في اكتشاف الخدمات المطلوبة.

لو أمعنا النظر في المكونات الثلاثة ، قد يتبادر إلى الذهن أن إضافة دلالات إلى كل من مستند الـ WSDL و مسجل الـ UDDI قد يمثل حلاً للمشكلة .

هذا الأمر يقودنا إلى جيل جديد من خدمات الويب يطلق عليها خدمات الويب الدلالية (Semantic Web Service) ، بشكل أدق خدمات الويب المفسرة بالدلالات الواضحة.

الفصل الخامس

1. لغة ترميز خدمات الويب OWL-S (Semantic Markup for Web Services)

كل خدمة ويب تكون موصفة من خلال الإجابة على بعض الأسئلة الشائعة العامة :

1. ما لذي تفعله الخدمة ؟ (What dose this web service do ?)
2. كيف تفعل ذلك الخدمة ؟ (How dose it do it ?)
3. كيف يتم استدعاء الخدمة ؟ (How is it invoked ?)

هذه الأسئلة لا علاقة لها بأي مجال خاص ولا تتطلب أي افتراضات لأي خدمة ويب معينة . وإذا نحن أنشأنا الانطولوجيا لتوفير المصطلحات التي يمكن أن تستخدم للإجابة على تلك الأسئلة العامة ، فإنها أيضاً ستكون انطولوجيا عامة ، كما أنها لن تكون ذات صلة إلى أي مجال محدد أو افتراضات معينة . هذا يختلف تماماً عن الانطولوجيا العادية ، التي هي دائماً مرتبطة بمجال معين . فإذا كانت الانطولوجيا المقدمة غير مرتبطة إلى أي مجال خاص و تحاول أن تصف صفوف و صفات عامة فإنها تدعى انطولوجيا عليا أو تأسيسية (Foundation or Upper Ontology) . حتى نرمز دلالات خدمات الويب وفقاً لما ذكر،لابد من توفر لغة ترميز خاصة على هذا المستوى.

تعتبر لغة الترميز الدلالي لخدمات الويب الـ OWL-S (Semantic Markup for Web Services) الموصاه من قبل W3C لوصف خدمات الويب واحدة من أكثر اللغات انتشاراً في هذا المضمار ، وهي مشتقة أساساً من لغة الـ OWL ، كما تعرف الصفوف والصفات التي ستستخدمها لوصف الجوانب الأساسية التالية :

- **ما لذي تفعله الخدمة ؟ (What dose this web service do ?)**

  تحتوي الـOWL-S على ما يدعى : Profile ontology (Profile.owl) لتعرف الصفوف والصفات التي تجيب على هذا النوع من الأسئلة . فهي بشكل رئيسي تستخدم للإعلان عن الخدمة ، لذلك تمكن الباحث عن الخدمة من تحديد فيما إذا كانت الخدمة المعطاة تقابل احتياجاته أو لا. يمكن لهذه الانطولوجيا الجزئية أن تجيب على فكرة اكتشاف الخدمة تلقائياً (Discovery).

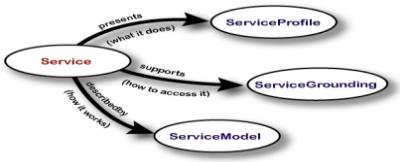
- **كيف تفعل ذلك الخدمة ؟ (How dose it do it ?)**

تحتوي الـ OWL-S على انطولوجيا جزئية (subOntology) تدعى : Process ontology (Process.owl) تعرف جميع المصطلحات التي تحتاجها لوصف كيف الخدمة تعمل ، بشكل أدق ، لوصف كيف ينبغي علينا أن نستخدم الخدمة من خلال وصف الإجراءات اللازم ة لتفاعل معها . يمكن من خلال هذه الانطولوجيا الجزئية الإجابة على فكرة تركيب الخدمات (Composition) ومراقبة تنفيذ العمليات (Monitoring) .

- **كيف يتم استدعائها ؟ (How is it invoked ?)**

تحتوي الـ OWL-S على انطولوجيا جزئية (subOntology) تدعى : Grounding ontology (Grounding.owl) تستخدم لتوفير المصطلحات التي تصف كيف يمكن الوصول إلى خدمة ما من الناحية تقنية ناجحاً ، كالتي تصف البروتوكول و تنسيقات الرسائل المتبادلة وعنوان الخدمة و المعالجة وأشكال أخرى ، فهي في الحقيقة توفر المواصفات الملموسة (Concrete) التي يمكن استخدامها في الاستدعاء التلقائي للخدمة (Invocation) .

تمثل الجوانب الثلاثة السابقة المكونات الرئيسية لـ OWL-S و معاً تشكل الهدف الرئيسي لهذه الانطولوجيا ، لكن لربط هذه الأجزاء مع بعضها تحوي الـ OWL-S إحدى أعلى مستويات الانطولوجيا تدعى Service ontology (Service.owl) لوصف كيف المكونات الثلاثة تعمل معاً لتصف بشكل كامل خدمة ويب . يمثل الشكل التالي عالم خدمات الويب كما هو مخطط له في لغة الـ OWL-S :



1.1. أين الدلالات التي تقدمها الـ OWL-S؟

الجواب ، في كل جانب من الجوانب الثلاثة المكونة للـ OWL-S هناك الصفوف (Classes) والصفات (Properties) والعلاقات (Relationship) والقيود (Structures) التي تقدم الدلالات الواضحة عن المعلومات التي تمتلكها الخدمة من مواصفات مجرد (Abstract) وملموسة (Concrete) التي تحقق الهدف من هذه الخدمة .

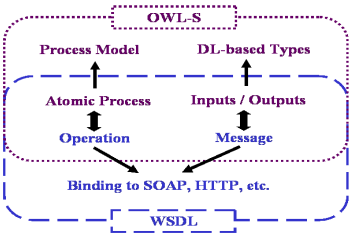
1.2. علاقة لغة الـ OWL-S و لغة الـ WSDL

قد نتساءل الآن إذا كانت الـ OWL-S تستخدم لوصف خدمات الويب ، وكانت لغة الـ WSDL تستخدم أيضاً لوصف خدمات الويب ، فما هي العلاقة بينهما ؟!

إن هذه العلاقة تنطوي على تكامل استخدام اللغتين معاً ، لإنشاء وصف متكامل لخدمات الويب ، فمن ناحية تستطيع لغة الـ OWL-S بما تحويه من تعابير لغة الـ OWL لأن تصف ما يعرف بالمواصفات المجردة (Abstract) مثل (الدخل ، الخرج ، العمليات) ، في الوقت الذي تستخدم فيه الـ WSDL الـ XML Schema التي لا تملك القدرة على الوصف من خلال صفوف الـ OWL .

بالمقابل الـ OWL-S ليس لديها وسيلة (حالياً) لأن تعبر عن المعلومات التقنية للخدمة (البروتوكول ، تنسيق الرسائل المتبادلة ، النقل) كما تفعل لغة الـ WSDL عبر المواصفات الملموسة (Concrete) .

هذا يقودنا إلى أن الـ OWL-S و WSDL متكاملان ، و لا يمكن أن يحل أحدهما مكان الآخر ليقدم صورة كاملة عن خدمة الويب . الشكل التالي يوضح العلاقة بينهما :



فكرة استخدام OWL-S مع WSDL هي تضمين تراكيب الـ OWL-S في مستند الـ WSDL من خلال تمديد عناصره التالية :

Type , Message , Operation , Binding

- Type : تسمح باحتواء مفردات من الـ OWL كيفية ، وذلك عبر تضمين هذا العنصر تلك المفردات بالشكل :

```
<types>
    <rdf :RDF namespace-declarations ... >
        ...............
    </rdf :RDF>
</types>
```

تشير النقاط إلى إمكانية إضافة ما نريد من تصريحات الـ OWL .

- Message : تسمح لمواصفات الـ Message Parts أن تمتلك صفوف (Classes) OWL كأنواع مجردة (Abstract Type) ، حيث فعلياً نشير إلى تطابق الـ Message Parts مع صفات (Properties) الدخل (Input) و الخرج (Output) في الـ OWL-S . وذلك بالشكل :

```
<message name="nmtoken">
    <part name="nmtoken" owl-s-parameter ="qname"/>
</message>
```

حيث يشير عنصر الـ owl-s-parameter إلى الـ XSD الذي يملك اسم مؤهل له ، وهو الذي يسمح بربط هذا الجزء بصفات (Properties) الدخل (Input) و الخرج (Output) في الـ OWL-S .

- Operation : نضع في هذا العنصر خاصية (Attribute) جديدة بهدف الإشارة إلى التطابق بين الـ Operation في مستند الـ WSDL و الـ Atomic Process في الـ OWL-S .

كما في شكل :

```
<operation name=" nmtoken "   owl-s-process=" nmtoken ">
```

تمثل الـ Atomic Process في الانطولوجيا الجزئية (subOntology) التي تدعى Process.owl التي تصف خدمة ويب على اعتبار أنها تتوقع رسالة واحدة وتعيد رسالة واحدة عند الاستجابة .

- Binding : نحن لا نمدد الـ WSDL ، وإنما نوفر نمط جديد للاستخدام مع SOAP Binding ، من خلال إضافة محدد URI للإشارة إلى الـ Encoding Style لكل Parts في عنصر الـ Message في مستند الـ WSDL ، و لتأكيد على استخدام تراكيب الـ OWL ، كأن نكتب :

http://www.daml.org/2001/03/

1.3. المطابقة من OWL-S إلى UDDI

مسجل الـ UDDI مصمم خصيصاً لاكتشاف خدمات الويب ، ولابد أن يتم تعزيزه ليقتني دلالات إضافية تساعد على رؤية الاكتشاف التلقائي لخدمات الويب ، هناك عدة طرق تستخدم في هذا الاتجاه إحداها إنشاء Mapping من الـ OWL-S إلى هيكل الـ UDDI ، عبر ربط عناصر مستند OWL-S-Profile بعناصر الـ UDDI على أساس واحد – لواحد . يمثل الـ tModel ، الصف الأول من تراكيب الـ UDDI ، الدور الرئيسي في هذه الفكرة من خلال إدراك القواعد الثلاثة التالية :

- القاعدة الأولى : لابد من وجود Interface منشأ ومسجل في الـ UDDI يضمن أن الخدمة الموجودة ستنفذ عليه . وهذا الـ Interface يكون ممثل من خلال tModel .
- القاعدة الثانية : مخططات التصنيف التي يتم استخدامها في تراكيب الـ UDDI لتصنيف الخدمات تكون ممثلة من خلال tModel قياسية منشأة ومحددة مسبقاً ، وبالتالي يمكن إضافة واحدة أو أكثر من هذه الـ tModel إلى الـ Interface الممثل أيضاً بالـ tModel حسب الحاجة لتسهيل البحث عن خدمة مطلوبة .
- القاعدة الثالثة : الـ tModels يمكن استخدامها لتمثيل الـ Namespace الذي يلعب دوراً هاماً في عملية الـ Mapping .

إن مرونة استخدام الـ tModel تقودنا إلى تعزيز الـ Interface بالمعلومات كأن نقدم للباحث عن الخدمة على سبيل المثال مدخلات (Inputs) ومخرجات (Outputs) الخدمة. الأمر الذي يساعد على تقرير ما إذا كانت الخدمة التي تطبق الـ Interface هي المطلوبة ، وذلك من خلال إضافة tModel إلى الـ Interface قياسية متفق عليها ، منعاً من حدوث اختلاف في أسماء الدخل (Input) والخرج (Output) التي قد يستخدمها المطورين ، كما هو الحال في إنشاء tModels لأنواع التصنيفات القياسية . والأمر الأكثر أهمية هو أن هذه الـ tModels ستستخدم كـNamespace التي لها الدور الكبير في الـ Mapping .

المثال التالي يوضح إنشاء tModel للدخل (Input) ندعوه بـ Input_tModel :

```
<tModel    tModelKey="uuid:E27972D8-717F-4516-A82D-B688DC70170C">
  <name> input_tModel </name>
  <description      xml: lang = "en">namespace of input_tModel </description>
  <overviewDoc>
          <description xml: lang="en">         whatever description you want
</description>
          <overviewURL>
                    http://www.ourCompany.com/internalDocuments/inputDefinition.html
          </overviewURL>
  </overviewDoc>
</tModel>
```

في عنصر < categoryBag > نحدد < keyedReference > لهذا الـ tModel :

```
<categoryBag>
          <keyedReference
                    tModelKey="uuid:E27972D8-717F-4516-A82D-B688DC70170C"
                    keyName="whatever-input-name-you-want"
                    keyValue="xsd:string"/>
<categoryBag>
```

من هنا علينا إدراك آلية الـ Mapping ، إذ تكمن الفكرة الرئيسية بإنشاء الـ tModels متخصصة في الـ UDDI لكل عنصر في مستند OWL-S-Profile لا يطابق عنصر في تركيب الـ UDDI . كأن ننشئ Input_tModel و Output_tModel متخصصة لعنصري الدخل والخرج في الـ OWL-S-Profile .

OWL-S Profile Elements	UDDI Elements
Input	BusinessService:categoryBag:input_tModel
output	BusinessService:categoryBag:output_tModel

في حين العناصر المتطابقة نطبق بينها مباشرة Mapping كما في معلومات الاتصال (Contact) ، وذلك كما في الجدول التالي :

OWL-S Profile Elements	UDDI Elements
contactInformation:phone	BusinessEntity:Contact:
contactInformation:email	BusinessEntity:Contact:email

قد يتطلب الأمر العديد من إنشاء tModels متخصصة ، لكن في نهاية ستكون النتيجة إضافة دلالات إلى مسجل الـ UDDI .

الفصل السادس

المقترح

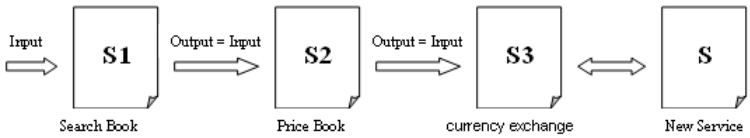
ما جمعناه في شرحنا سيقودنا مباشرة نحو الهدف الحقيقي الذي نسعى إليه ، وهو بناء محرك بحث ذكي قادر على فهم المعلومات المرمزة دلالياً عبر لغة ترميز خدمات الويب OWL-S بما تحويه من مفردات لغة OWL أو لغة أخرى .

ليكون العامل الحقيقي وراء الاكتشاف التلقائي لخدمات الويب المرمزة دلالياً واستدعائها وتركيب خدمات جديدة آلياً من خدمات ويب متوفرة و مرمزة دلالياً .

على سبيل المثال نريد بحث عن خدمة (***S***) تسمح لنا بالحصول على سعر كتاب ما بالعملة المحلية ، لكنها خدمة غير متوفرة ، في حين يمكن تركيبها من الخدمات التالية :

1. خدمة البحث عن كتاب (***S1***) تقدم لنا معلومات عن كتاب ما (اسم المؤلف ، تاريخ إصدار الكتاب ، اسم الناشر ، رقم التسلسلي ISBN) والتي تمثل خرج الخدمة (Output) ، وذلك من خلال تدوين عنوان الكتاب كدخل (Input) للخدمة.
2. خدمة الحصول على سعر كتاب عبر موقع بيع كتب (***S2***) تقدم لنا سعر كتاب ما كخرج للخدمة (Output) من خلال تدوين الرقم التسلسلي للكتاب ISBN كدخل للخدمة (Input) .
3. خدمة تحويل العملات (***S3***) .

من خلال دمج الخدمات الثلاثة السابقة ، تتركب لدينا خدمة جديدة قادرة على تحقيق ما نرغب إليه من نتائج فعالة وذات دقة عالية . الشكل التالي يوضح آلية التركيب :



(References) المراجع

- www.w3.org/RDF
- www.w3schools.com/rdf
- www.w3.org/TR/rdf-schema/
- www.w3schools.com/rdf/rdf-schema
- www.w3.org/TR/owl-features
- http://jena.sourceforge.net
- http://protege.standford.edu
- semantic web programming 2009 (John Hebeler , Mathhew Fisher , Rayn Blace ,Andrew Perez-Lopez)
- Semantic Web for Dummies 2009 (Jeffery T.Pollock)
- Web Services in 24 Hours 2003 (Stephen ptts , Mike Kopack)
- www.w3schools.com/webservices
- www.w3.org/submission/owl-s
- Adding OWL-S to UDDI, implementation and throughput (Naveen Srinivasan, Massimo Paolucci , Katia Sycara)
- Importing The Semantic Web in UDDI 2002 (Massimo Paolucci , Takahiro Kawamura , Terry R.Payne , Katia Sycara)