# 시맨틱 웹 어플리케이션의 빠른 개발을 위한 프레임워크

김탁은 O 김명호

KAIST 전산학과

tekim@dbserver.kaist.ac.kr, mhkim@dbserver.kaist.ac.kr

# A Framework for Supporting Rapid Development of the Semantic Web Application

Tak-Eun Kim O Myoung Ho Kim

Division of Computer Science, KAIST

요 약

최근 시맨틱 웹에 대한 많은 관심이 집중되고 있다. 그러나 시맨틱 웹 기술의 핵심이 되는 온톨로지 개발의 어려움과, 정형화된 개발 방법의 부재로 인해 일반 개발자들이 시맨틱 웹 응용프로그램을 개발하는 데에 많은 어려움을 겪고 있다. 본 연구에서는 온톨로지와 관련 기능들을 컴포넌트화 하고, 컴포넌트들을 결합하여 응용프로그램을 쉽고 빠르게 구현하기 위한 프레임워크를 제안한다. 이는 복잡한 시맨틱 웹 기술을 숨김으로써, 일반 개발자들이 시맨틱 웹에 대한 깊은 이해 없이도 시맨틱 웹 응용프로그램을 쉽고 빠르게 구현할 수 있도록 한다.

## 1. 서론

### 1.1. 동기

웹에서 의미 있는 정보들을 추출하여, 이를 정보 처리 및 지식기반 서비스에 활용하려는 연구들이 수행되어 왔다. 시맨틱 웹 기술은 이러한 노력들 중 하나로, 웹 상의 여러 소스로부터 얻어온 다양한 형태의 데이터들을 체계적으로 통합하고 처리하는 데에 특화된 기술로 각광을 받고 있다. 특히 데이터 모델에 의미(Semantic)를 함께 기술할 수 있어, 컴퓨터가 의미 기반의 정보처리를 가능하게 한다 [1].

학계 및 산업분야에서 RDF 또는 OWL과 같은 시맨틱 웹 표준 기술을 이용한 많은 연구들이 진행되어왔고, 최근 들어 상용 서비스들도 점차 개발되고 있다. 예로 Semantic MediaWiki [2,3]는 시맨틱 웹 기술을 이용한 협업 문서작성 도구로, "London isLocatedIn England"와 같이 RDF로 각 문서에 나타난 개체들간의 관계를 기술할 수 있도록 하여, 컴퓨터가 이러한 정보를 바탕으로 의미 기반의 정보처리를 할 수 있도록 한다. NEPOMUK[1]은 개인 컴퓨터에 저장된 정보를 관리하고, 이를 다른 사람과 공유하기 위한 Social Semantic Desktop이다. 그밖에 시맨틱 블로깅을 위한 도구나, 시맨틱 검색 엔진 등과 같은 다양한 시맨틱 응용 프로그램들이 있다 [4].

하지만 이러한 시맨틱 웹 응용프로그램을 만드는 데에는 다음과 같은 문제점들이 있다.

첫째로, 시맨틱 웹의 핵심이 되는 온톨로지 설계가 쉽지 않다는 점이다. 온톨로지를 잘 설계하기 위한 가이드라인들이 많이 제안되었지만 [5,6], 가이드라인에 명시된 디자인 방법들은 간단한 구조의 온톨로지에 적용 가능한 경우가 많아 실제 설계에 적용하기가 어렵다. 이러한 문제점과 함께 시맨틱 웹 응용프로그램 개발을 위한 문서나 각종 도구 부족으로 인해 개발자들에게 온톨로지 기반의 응용프로그램 개발이 매우 어려운 일로 인식되고 있다.

둘째로, 각 온톨로지 개발자들마다 디자인 철학이 다르기 때문에, 동일한 객체에 대해서도 매우 다른 구조로 온톨로지를 설계할 수가 있다. 예를 들어 SKOS[2]와 같은 시소러스 온톨로지를 구축하기 위해서는 용어(term)를 나타내는 클래스와, 용어들 간의 계층 정보를 나타내기 위한 broader라는 이름의 프로퍼티가 필요하다. 여기서 한 온톨로지 개발자는 broader 프로퍼티의 의미를 "The term A is a broader term of B"라고 정의하고, 다른 온톨로지 개발자는 "The term A has a broader term B"라고 정의할 수 있다. 즉, 동일한 이름의 온톨로지 용어(vocabulary)라고 할지라도, 그 의미는 매우 다르게 사용될 수 있다. 이러한 디자인의 다양성 문제는 온톨로지 데이터가 여러 사람들에 의해 공유될 때 매우 심각한 문제가 될 수 있다. 따라서 기존 온톨로지를 최대한 재사용 할 수 있는 방법이 필요하다.

소프트웨어 라이브러리는 재사용 가능한 코드의

[1] Networked Environment for Personal Ontology-based management of Unified Knowledge

[2] Simple Knowledge Organization System

묶음으로, 개발자는 내부 구조를 알 필요 없이 라이브러리를 조합하여 소프트웨어를 쉽고 빠르게 구현할 수 있다 [7]. 이와 같이 온톨로지 전문가에 의해 잘 설계된 온톨로지가 있고, 일반 개발자들은 이를 조합하여 응용프로그램을 만들 수 있다면, 개발자는 시맨틱 웹에 대한 깊은 이해 없이도 시맨틱 웹 응용프로그램을 쉽고 빠르게 개발할 수 있을 것이다. 본 연구에서는 이를 위한 프레임워크를 제안한다.

### 1.2. 방법

본 연구에서는 온톨로지 전문가에 의해 잘 설계된 온톨로지, 온톨로지를 처리하는 코드 등을 패키징하여 컴포넌트로 만들고, 이들을 조합하여 시맨틱 응용프로그램을 쉽게 만들 수 있는 프레임워크를 제안한다. 제안하는 프레임워크를 통해 일반 개발자들은 시맨틱 웹에 대한 깊은 이해 없이 다양한 종류의 시맨틱 웹 응용프로그램을 쉽고 빠르게 제작할 수 있다. 예를 들면, MS Outlook과 같이 개인 정보를 기록, 관리하기 위한 시맨틱 웹 응용프로그램을 만든다고 할 때, 모든 기능을 처음부터 새로 만들지 않고, 일정관리 컴포넌트, 이메일 컴포넌트, 연락처 컴포넌트 등을 결합하여 쉽게 만들 수가 있다.

본 연구에서는 앞서 논의한 컴포넌트를 온틀릿(Ontlet) 이라고 부른다. 온틀릿은 MVC(Model-View-Controller)모델과 매우 유사한데, 데이터 모델로 온톨로지를 가지며, 온톨로지 인스턴스를 생성, 편집, 표현하기 위한 뷰, 뷰에서 발생되는 이벤트를 처리하기 위한 컨트롤러로 구성된다. 온틀릿은 모델, 뷰, 컨트롤러를 모두 갖추고 있으므로 특정 기능만을 수행하는 작은 시맨틱 웹 응용프로그램으로서 단독적으로 동작할 수도 있으며, 규모가 큰 시맨틱 웹 응용프로그램 구축을 위해 다른 온틀릿과 결합할 때에는 라이브러리로 쓰이기도 한다.

본 연구의 기여점은 다음과 같다. 첫째로, 온톨로지와 관련 기능들을 컴포넌트화 하고, 이를 결합하는 방법을 통해 시맨틱 웹 응용프로그램을 쉽고 빠르게 개발할 수 있다. 둘째로, 온틀릿을 통해 시맨틱 웹의 복잡한 부분을 숨김으로써 시맨틱 웹 기술을 잘 모르는 일반 개발자들도 시맨틱 웹 응용프로그램 개발을 가능하도록 함으로써, 시맨틱 웹 응용프로그램 개발을 활성화 할 수 있다.

논문의 나머지 부분은 다음과 같이 구성된다. 2장에서는 시맨틱 웹 응용프로그램의 개발에 관한 관련 연구에 대해서 논의한다. 3장에서는 온틀릿 모델에 대해서 상세히 논의하고, 4장에서는 온틀릿의 각 구성요소들이 어떻게 결합하여 하나의 응용프로그램을 구성할 수 있는지 그 방법에 대해서 구체적으로 논의한다. 5장에서는 프레임워크 구현에 대한 내용을 다루고, 6장에서 제안하는 프레임워크의 기여점을 논의하고, 향후 연구에 대해서 논의한다.

## 2. 관련 연구

시맨틱 웹 응용프로그램의 개발에 관한 많은 내용들이 Semantic Desktop 분야에서 논의되었다. Leo Sauermann이 처음으로 제안한 Semantic Desktop [8]은 개인 컴퓨터에 저장된 수많은 데이터들을 시맨틱 웹 기술을 써서 관리하는 것이 목적이다. 모든 데이터들이 RDF로 기술됨으로써, 응용 프로그램 간에 데이터를 표준화된 방법으로 쉽게 공유할 수 있다는 것이 장점이다.

NEPOMUK 프로젝트[9]는 Leo Saurmann의 후속 연구로, 이는 Semantic Desktop에 소셜 네트워크 기능을 추가한 연구이다. NEPOMUK 프로젝트는 IBM, SAP과 같은 많은 기업들의 지원의 받고 있는 대형 프로젝트로, 그 기반 구조가 체계적으로 잘 설계되어 있다. 하지만 NEPOMUK은 Semantic Desktop에 초점을 맞춰 설계되었기 때문에, 설계한 온톨로지 역시 개인 컴퓨터에 저장된 리소스를 다루는 데에 특화되어 있다. 따라서 Semantic Desktop이 아닌 다른 시맨틱 웹 응용프로그램 개발에 NEPOMUK 프로젝트의 결과물을 곧바로 적용하기에는 어려운 점이 많다. 설계된 온톨로지 역시 매우 복잡하여 외부 온톨로지를 쉽게 추가할 수 없으며, 개발자들이 복잡한 온톨로지 구조를 이해하는 데에 많은 노력이 요구된다는 단점이 있다.

Cruz [10]는 Semantic Desktop 설계를 위한 계층화된 프레임워크를 제안하였다. 온톨로지를 모듈화하고, 이를 또다시 계층화 함으로써 온톨로지의 관리가 용이하고, 외부 온톨로지를 쉽게 통합할 수 있다는 장점이 있다. 그 밖에 온톨로지를 디자인하기 위한 PIA designer나, 온톨로지 인스턴스를 탐색하기 위한 Resource Browser와 같은 사용자 인터페이스를 갖추고 있지만, 이들의 표현력이나 기능들이 매우 제한적이다.

Haystack[11]은 사용자의 라이프로그를 관리하기 위한 시스템이다. 다른 연구들과 차별화 되는 점으로 Ozone이라는 사용자 인터페이스 디자인을 위한 언어를 제공하여 쉽게 사용자 인터페이스를 구성할 수 있다. 하지만 Ozone으로 표현할 수 있는 사용자 인터페이스의 형태가 매우 제한적이고, 웹 표준 언어가 아니기 때문에 서드파티 응용프로그램과의 통합이 매우 어렵다. 또한 데이터 모델로 사용되는 온톨로지 역시 확장이 용이하지 않아, 외부 온톨로지와 통합이 어렵다.

## 3. 온틀릿 모델

온틀릿은 온톨로지, 온톨로지 인스턴스를 생성, 표현, 수정 위한 뷰, 뷰에서 발생한 사용자 액션을 처리하기 위한 컨트롤러로 구성된다. 3장에서는 온틀릿의 세가지 컴포넌트에 대해서 각각 살펴보도록 한다.

### 3.1. 온틀릿 데이터 모델: 온톨로지

그림 1은 온틀릿의 데이터 모델 스키마를 나타내고 있다. 모든 온틀릿들은 그림 1에 정의된 형태의

온톨로지를 데이터 모델로 가져야 한다. 그림 1에서 타원형은 OWL 클래스, 사각형은 리터럴, 화살표는 프로퍼티를 나타낸다. 화살표의 시작점에 있는 클래스는 프로퍼티의 domain이고, 화살표가 가리키는 곳의 리터럴 또는 클래스는 프로퍼티의 range를 나타낸다.

각 용어들을 살펴보면 다음과 같다. ontlet:OntletHead 클래스는 온틀릿을 대표하는 클래스로, 각 온틀릿에 오직 한 개만 존재한다. ontlet:DataObject 클래스는 응용프로그램이 사진과 같은 바이너리 데이터를 저장할 필요가 있을 때, 바이너리 데이터가 저장된 경로 및 메타데이터를 저장하기 위해 정의한 클래스이다. 예를 들어 다음과 같이 사용될 수 있다.[3]

p:John rdf:type p:Person .
p:John p:faceImage p:JohnFaceImage .
p:JohnFaceImage rdf:type ontlet:DataObject .
p:JohnFaceImage ontlet:path "http://.../john.jpg" .

ontlet:reference 프로퍼티는 다른 온틀릿 데이터 모델과 연결하기 위해 사용된다. 그림 2에서 볼 수 있듯이 ontlet:reference 프로퍼티는 한 온틀릿의 임의 클래스와 다른 온틀릿의 ontlet:OntletHead 클래스를 연결한다. ontlet:reference를 통해 여러 온틀릿의 온톨로지들을 연결함으로써 구현하려는 응용프로그램에 적합한 형태의 온톨로지를 구축할 수 있게 된다.

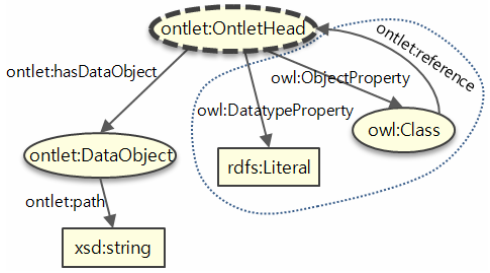


그림 1. 온틀릿 데이터 모델

그림 1에서 점선으로 묶인 부분은 온톨로지 개발자가 임의로 클래스나 프로퍼티를 정의할 수 있는 부분이다.

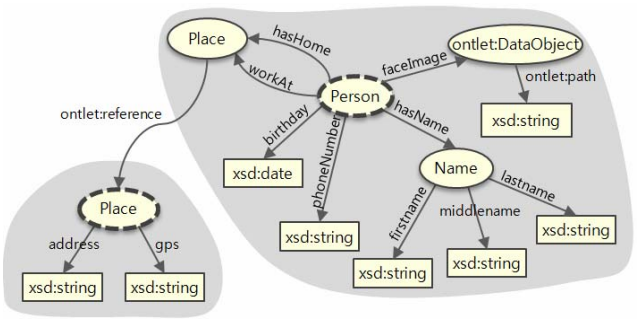


그림 2. Person 온틀릿 (오른쪽)과 Place 온틀릿 (왼쪽)

그림 2는 온틀릿 데이터 모델을 바탕으로 구축한 온톨로지의 예를 보여주고 있다. 그림 2에서는 Person과 Place라는 두 개의 온틀릿 온톨로지가 있으며, 이를 ontlet:reference 프로퍼티로 연결함으로써, 아래와 같이 확장된 온톨로지를 구성하였다.

### 3.2. 온틀릿 뷰

일반적으로 웹 응용프로그램은 사용자로부터 새로운 데이터를 입력 받기 위한 create 뷰, 데이터베이스에 저장된 정보를 보여주기 위한 display 뷰, 기존 데이터를 수정하기 위한 modify 뷰를 가지고 있다. 이와 동일하게 온틀릿도 위에서 언급된 세가지 종류의 뷰를 가진다. Create 뷰를 통해서 사용자는 온틀릿의 온톨로지 인스턴스를 생성할 수 있고, display 뷰를 통해 온톨로지 인스턴스를 사용자에게 친화적인 방법으로 보여줄 수 있다. 그리고 modify 뷰를 통해 기존에 입력된 온톨로지 인스턴스를 수정할 수 있다.

온틀릿 뷰를 기술하기 위해서, 본 연구에서는 OntML(Ontlet Markup Language)라는 XML 기반의 뷰 기술 언어를 설계하였다. 그림 3은 OntML로 기술된 온틀릿 뷰의 일부분을 보여주고 있다.

OntML은 HTML에 몇 가지 문법을 추가한 것으로, XML 헤더와 스크립틀릿(scriptlet)을 제외하면 HTML과 동일하다. 스크립틀릿은 %{와 }의 기호 사이에 기술되는데, 다음 장에서 논의될 컨트롤러에 의해 해석된 후, 컨트롤러가 생성된 값으로 치환된다.

다음의 예를 보자. 그림 3은 그림 2의 Person 온틀릿 온톨로지의 인스턴스를 보여주기 위한 display 뷰이다. %{ val(&base;firstname) } 스크립틀릿의 경우, 컨트롤러는 우선 온틀릿 온톨로지에서 해당 프로퍼티를 가지는 RDF triple을 찾아내고, 스크립틀릿을 이 RDF triple의 리터럴 값으로 치환하게 된다. 이와 같은 방법으로 나머지 스크립틀릿도 값으로 치환하게 되면 사용자는 적절한 값들로 채워진 display 뷰를 볼 수 있게 된다.

그밖에 %{ importView() } 스크립틀릿은 여러 온틀릿의 뷰를 통합하기 위해 사용된다. 예를 들면, 그림 2에서 Person 온틀릿이 Place 온틀릿을 포함하고 있기 때문에, Person 온틀릿 뷰에 Place 온틀릿 뷰가 포함되어야 한다. 이러한 뷰 통합 과정을 importView() 스크립틀릿을 통해 쉽게 해결할 수 있다. 뷰 통합에 대해서는 4.2절에서 자세히 논의한다.

```
<?xml version="1.0"?>
<!DOCTYPE view [
    <!ENTITY base "http://www.ontlet.org/ontlet/person/" >
]>
<view>
  <![CDATA[
  <table>
      <tr>
        <td>Name</td>
        <td>%{ val("&base;firstname") + " " +
```

[3] 표기의 편의상 리소스의 prefix를 p로 나타내었다.

```
                    val("&base;middlename") + " " +
                    val("&base;lastname") }</td>
            </tr>
            ....
            <tr>
                <td>Home</td>
                <td>%{ importView("&base;hasHome") }</td>
            </tr>
            ...
    </table>
    ]]>
</view>
```

그림 3. Person 온틀릿의 display 뷰 코드

본 연구에서 제안하는 프레임워크에서는 개발자의 편의를 위해 온톨로지로부터 그림 3과 같은 뷰 코드를 자동 생성하는 도구를 제공한다. 앞서 설명하였듯이 OntML은 스크립틀릿만 제외하면, HTML과 동일하다. 따라서 기존 웹 개발에 익숙한 개발자들이 시맨틱 웹 응용프로그램 개발에 쉽게 참여할 수 있고, 구글맵, 플래쉬 기반의 위젯과 같은 서드파티 응용프로그램을 쉽게 추가할 수 있다.

### 3.3. 온틀릿 컨트롤러

온틀릿 컨트롤러는 3.2절에서 설명한 세가지 종류의 뷰에서 발생하는 이벤트를 처리하는 역할을 담당한다. 이러한 이벤트들은 온톨로지 인스턴스를 생성, 요청, 변경, 삭제하는 작업들 중 하나이므로, 컨트롤러는 네 가지 작업을 처리할 수 있어야 한다.

한 온틀릿은 최소 한 개의 컨트롤러를 가져야만 한다. 온틀릿 개발자들이 온틀릿을 하나 개발할 때마다 컨트롤러를 구현해야 한다면 매우 번거로울 것이다. 다행이 많은 온틀릿들이 사용자가 뷰에 입력한 내용을 온톨로지 인스턴스로 저장하거나, 사용자의 요청에 맞는 온톨로지 인스턴스를 찾아서 보여주는 등의 기본적인 작업을 수행한다. 따라서 프레임워크에 기본적인 작업 처리를 담당하는 범용 컨트롤러를 포함시켜 이를 이용하게 함으로써, 기본 작업들만 필요한 온틀릿들은 개발자가 컨트롤러를 따로 구현하지 않아도 된다.

하지만 특별한 기능이 필요한 온틀릿의 경우에는 해당 컨트롤러를 직접 구현하여야 한다. 예를 들어 사진 관리를 위한 Photo 온틀릿을 만든다고 하자. 사용자가 사진 파일을 업로드 했을 때 EXIF 정보를 추출하여 온톨로지에 저장하고자 한다면, 이는 범용 컨트롤러에서 처리할 수 없으므로 Photo 온틀릿 컨트롤러가 EXIF 정보를 추출할 수 있도록 컨트롤러를 직접 구현하여야 한다.

## 4. 온틀릿 결합

4장에서는 3장에서 논의한 온틀릿의 세가지 구성요소가 다른 온틀릿의 구성요소들과 어떻게 결합하는지 살펴보고, 온틀릿들을 결합하여 어떻게 시맨틱 웹 응용프로그램을 개발할 수 있는지에 대해서 살펴본다.

### 4.1. 온틀릿 간의 온톨로지 결합

각 온틀릿의 데이터 모델(온톨로지)을 통합하는 방법은 매우 간단하다. 그림 2와 같이 ontlet:reference 프로퍼티를 이용하여 각 온틀릿의 온톨로지를 통합하면 된다. 따라서 이미 만들어져 있는 온틀릿들의 온톨로지들을 통합하여, 복잡한 시맨틱 웹 응용프로그램 온톨로지를 쉽고 빠르게 구축할 수 있다.

### 4.2. 온틀릿 간의 뷰 통합

Person 온틀릿과 Place 온틀릿을 결합하여 간단한 Contact 응용프로그램을 만드는 예제를 통해 두 온틀릿의 뷰를 어떻게 통합하는지 살펴보자.[4]

그림 4는 Person 온틀릿과 Place 온틀릿의 display 뷰를 각각 나타내고 있다. 이 두 온틀릿의 온톨로지는 그림 2에 나타나 있고, Person 온틀릿의 display 뷰에 대한 OntML 코드는 그림 3에 기술되어 있다.

그림 2에서 Person 온틀릿 온톨로지에서 Place 온틀릿 온톨로지를 hasHome과 workAt 프로퍼티로 연결하여, 집과 직장의 위치 정보를 기록하는 데에 사용하려는 것을 볼 수 있다. 그리고 그림 3에서, Person 온틀릿의 display 뷰가 importView() 스크립틀릿을 이용해 Place 온틀릿의 display 뷰를 로드하려는 것을 볼 수 있다.

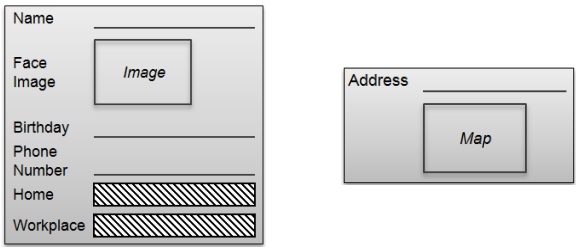


그림 4. Person 온틀릿의 display 뷰 (왼쪽)와 Place 온틀릿의 display 뷰 (오른쪽)

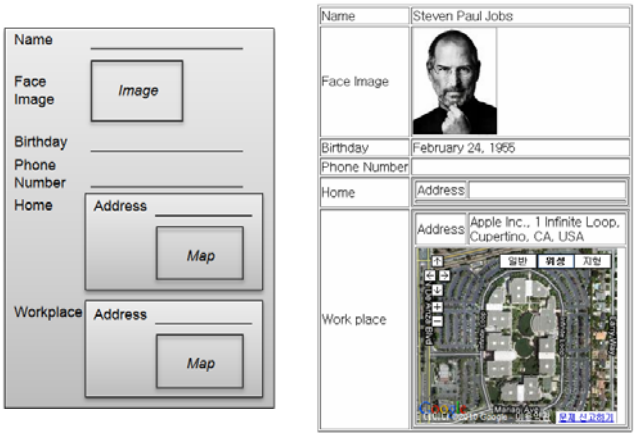


그림 5. 그림 4의 두 뷰가 통합된 결과

Place 온틀릿의 뷰가 로드될 자리는 그림 4에서

[4] 두 온틀릿의 온톨로지는 그림 2에 나타나 있다.

빗금친 두 부분인데, importView()를 통해 Place 온틀릿 뷰가 로드되어 통합되면 그림 5의 왼쪽과 같은 통합된 뷰가 생성된다. 오른쪽은 실제 구현된 프레임워크에서 두 뷰를 결합한 예이다.

그림 5와 같이 여러 온틀릿들의 뷰가 하나로 통합되면, 사용자는 통합된 뷰의 코드를 수정하여 사용자 친화적인 인터페이스로 만들 수 있다. 3.2절에서 설명하였듯이, 온틀릿 뷰는 스크립틀릿과 헤더를 제외하고는 모두 HTML로 기술되어있기 때문에, 서드파티 응용프로그램을 자유롭게 가져다 쓸 수 있다. 따라서 그림 5과 같이 Place 온틀릿의 GPS값을 구글맵에 표현하여 사용자에게 더욱 친숙한 방법으로 보여줄 수 있다.

### 4.3. 온틀릿 간의 컨트롤러 통합

사용자는 4.2절에서 논의한 통합 뷰와 직접 상호작용을 한다. 통합 뷰는 여러 온틀릿의 뷰가 결합되어 생성된 것이므로, 통합 뷰에서 생성된 이벤트를 처리하기 위해서는 각 온틀릿의 컨트롤러 역시 통합되어야 한다.

프레임워크는 통합 뷰에서 뷰의 각 부분이 어느 온틀릿으로부터 나온 것인지를 파악하고 있다. 따라서 사용자가 통합 뷰와 상호작용을 통해 이벤트를 발생시키면, 프레임워크는 이벤트 처리에 필요한 일들을 분할하여 해당 온틀릿 컨트롤러가 이를 처리하도록 한다.

예를 들어 앞에서 예로 보인 Contact 응용프로그램을 생각해보자. 사용자가 새로운 친구 한명을 추가하려고 한다고 하자. 사용자는 친구의 이름, 생일, 전화번호, 친구의 집 또는 직장 주소를 create 뷰를 통해 입력할 것이다. 그림 2를 보면, 이름, 생일, 전화번호는 Person 온틀릿이 관할하고, 집 또는 직장 주소 정보는 Place 온틀릿이 관할한다는 것을 알 수 있다.

프레임워크는 이러한 사실을 알고 있어서 사용자가 뷰에서 모든 입력을 마친 후 저장 버튼을 누르게 되면, 이름, 생일, 전화번호에 대한 RDF triple 생성은 Person 온틀릿 컨트롤러에게, 집 또는 직장 주소에 대한 RDF triple 생성은 Place 온틀릿 컨트롤러에게 처리하게 한다.

우선, Person 온틀릿 컨트롤러는 다음과 같은 RDF triple을 생성한다.

pe:Person1 rdf:type pe:Person .
pe:Person1 pe:firstname “Steven” .
pe:Person1 pe:middlename “Paul” .
pe:Person1 pe:lastname “Jobs” .
pe:Person1 pe:birthday “February 24, 1955” .

주소 정보는 Place 온틀릿이 관할하므로, Place 온틀릿 컨트롤러는 다음과 같은 RDF triple을 생성한다.

pl:Place1 rdf:type pl:Place .
pl:Place1 pl:address “Apple Inc., … USA” .
pl:Place1 pl:gps “N 37° 19' 55" W 122° 1' 48"” .

그리고, 각 온틀릿에서 생성된 RDF triple을 연결하기 위해 프레임워크는 추가적으로 다음과 같은 RDF triple을 생성한다.

pe:Person1 pe:workAt pl:Place1 .

프레임워크는 이벤트 처리에 관여하는 온틀릿들을 순차적으로 호출하고, 이를 연결함으로써 컨트롤러들을 통합한다.

### 4.4. 정리

4장에서 논의한 내용을 간략하게 정리하면 그림 6과 같다. 복잡한 응용프로그램을 만들기 위해서는 각기 고유한 기능을 가진 여러 온틀릿들을 결합하여 만들 수 있는데, 프레임워크는 각 온틀릿의 여러 뷰와 컨트롤러, 그리고 데이터 모델인 온톨로지들을 통합하여 결합된 뷰, 결합된 컨트롤러, 결합된 온톨로지를 생성할 수 있게 해준다. 응용프로그램은 마치 하나의 컨트롤러와 데이터 모델, 뷰를 가진 것처럼 보이지만, 실은 여러 온틀릿들로부터 나온 것들이며, 프레임워크가 여러 온틀릿들을 유기적으로 연결하여 데이터를 처리할 수 있는 방법을 제공한다.

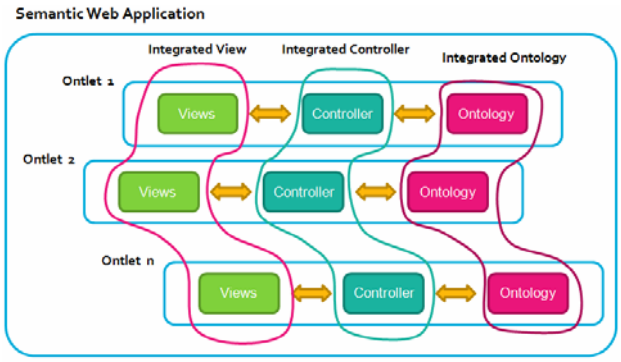


그림 6. 온틀릿의 통합과 그 의미

## 5. 구현

제안한 프레임워크는 JAVA로 구현되었으며, 온톨로지 처리 엔진으로 Jena와 TDB를 사용한다. 프레임워크는 JAR (Java Archive) 파일로 패키징되어 있고, API를 통해 프레임워크의 기능을 사용할 수 있다. 각각의 온틀릿은 온톨로지와 뷰, Java로 구현된 라이브러리가 정해진 디렉토리 구조로 패키징되어 Ontlet Storage에 저장되어 있다.

시멘틱 웹 응용프로그램을 만들기 위해 개발자는 Servlet 프로그래밍을 하게 되는데, 프레임워크가 Servlet에 로드되고, 개발자는 프레임워크 API를 통해 각종 온틀릿들을 불러와서 통합하고, 통합 뷰를 편집하여 사용자에게 보기 좋게 수정한 다음, 이를 Servlet으로 실행하면 된다. 그림 5의 오른쪽이 Person 온틀릿과 Place 온틀릿을 통합해 간단한 Contact

응용프로그램을 만든 예를 보여주고 있다.

프레임워크의 활성화를 위해서는 풍부한 온틀릿이 필수적이다. 따라서 본 연구에서는 Video, Audio, Image 등과 같은 기본적인 온틀릿들을 프레임워크에 미리 설계해두었다. 본 프레임워크는 아직 개발 중으로, 차후에 다양한 온틀릿들을 늘려갈 계획이다.

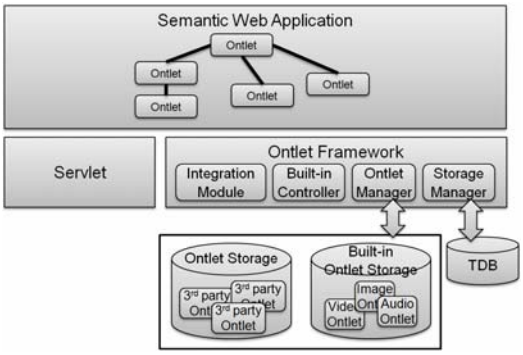


그림 7. 프레임워크의 구조

## 6. 결론 및 향후 연구

시맨틱 웹 기술은 많은 주목을 받고 있지만 기존의 패러다임을 새롭게 재편하는 부분들이 많아 많은 사람들이 기술을 이해하는 데에 어려움을 겪고 있다. 이러한 문제는 시맨틱 웹 응용프로그램 개발을 활성화하는 데에 걸림돌이 되고 있다.

소프트웨어 개발에서, 모듈화된 디자인 방법론은 개발자들이 모든 세부 내용을 구현할 필요 없이 쉽고 빠르게 소프트웨어를 개발할 수 있도록 도와준다.

본 연구에서는 이러한 방법론을 시맨틱 웹 응용프로그램 개발에 도입하였다. 온톨로지 전문가에 의해 설계된 온톨로지와 이를 다루는 데에 필요한 코드를 함께 묶어 컴포넌트화 한 온틀릿을 만들고, 여러 온틀릿들을 조합하여 시맨틱 웹 응용프로그램을 쉽고 빠르게 만들 수 있는 방법을 제안하였다. 더욱이 프레임워크의 대부분이 Servlet과 AJAX와 같이 기존에 잘 알려진 웹 기술을 사용하고 있다. 따라서 시맨틱 웹에 대한 깊은 이해가 없는 일반 개발자들이 다양한 종류의 시맨틱 웹 응용프로그램을 쉽고 빠르게 제작할 수 있도록 하였다.

본 연구에서 제안한 프레임워크는 한창 개발이 진행 중이다. 따라서 다음과 같이 많은 문제들을 향후 연구로 남겨두고 있다. 우선 외부 온톨로지를 사용할 때에 온틀릿 데이터 모델로 변경을 해야 다른 온틀릿과 통합이 가능한데, 향후 연구에서는 외부 온톨로지를 이러한 변경 없이 다른 온틀릿과 통합할 수 있는 방법에 대해서 연구할 계획이다. 또한 본 연구에서는 모든 데이터들을 온톨로지 인스턴스로 저장하고 있는데, 관계형 데이터베이스와 결합하여 메타데이터만 온톨로지로 저장하고, 나머지 정보들은 관계형 데이터베이스에 저장하는 방법에 대해서도 연구할 계획이다. 그리고 프레임워크의 유용성 평가도 수행할 계획이다.

## 7. Acknowledgement

이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2009-0083055).

## 8. 참고 문헌

[1] A. Maedche, B. Motik, L. Stojanovic, R. Studer and R. Volz, "Ontologies for Enterprise Knowledge Management", IEEE Intelligent Systems, Vol. 18, No. 2, pp. 26-33, 2003.

[2] M. Krötzsch, D. Vrandecic and M. Völkel, "Semantic MediaWiki", LNCS, Vol. 4273, pp. 935-942, 2006.

[3] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller and R. Studer, "Semantic Wikipedia", Proc. of WWW2006, pp. 585-594, 2006.

[4] J.T. Pollock, "Semantic Web for Dummies", Wiley Publishing Inc.

[5] N.F. Noy and D.L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology", Technical Report SMI-2001-0880, Stanford Medical Informatics, 2001.

[6] G. Goos et al., "Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, LNCS 5445, 2009.

[7] I. Crnkovic, "Component-based software engineering - new challenges in software development", Software Focus, Vol. 2, Issue 4, pp. 127-133, 2002.

[8] L. Sauermann, "The gnosis-using semantic web technologies to build a semantic desktop", Deploma thesis, Technical University of Vienna, 2003.

[9] T. Groza, S. Handscheuh and K. Möller, "The NEPOMUK Project - On the Way to the Social Semantic Desktop", Proc. of I-SEMANTICS2007, Graz, Austria, 2007.

[10] I.F. Cruz and H. Xiao, "A layered framework supporting personal information integration and application design for the semantic desktop", VLDBJ, Vol. 17, No. 6, pp. 1385-1406, 2008.

[11] D. Quan, D. Huynh and D.R. Karger, "Haystack: A Platform for Authoring End user Semantic Web Applications", LNCS, Vol. 2870, pp. 738-753, 2003.