특집 07

확장 필드 응용으로 테넌트별 설정이 가능한 애플리케이션 서비스 시스템

오병택·원희선·허성진 (한국전자통신연구원)

목 차 »

1. 서 론

클라우드 컴퓨팅 기술을 기반으로 제공되는 서비스는 Robert Anderson's의 3 layers에서 주장하는 바에 따르면 크게 IaaS, PaaS 그리고, SaaS와 같은 3개의 layer로 말할 수 있다[1]. 3개의 layer중 IaaS는 하부의 하드웨어와 데이터 저장소 등의 컴퓨터 시스템 자원을 서비스로 제공하고, PaaS는 이를 기반으로 애플리케이션 개발환경과 실행 기반을 제공하며, SaaS 서비스는 IaaS와 PaaS를 기반으로 실행되는 애플리케이션을 서비스로 제공한다. 종래 애플리케이션 제공 방식은 사용자의 단말에 직접 설치하는 모델이나 애플리케이션을 온라인으로 서비스하는 ASP 모델로, 대부분 애플리케이션 하나당 사용 기관 하나로 실행해서 최초 설치시에 기관에 맞게 설정하는 커스터마이즈 요소가 많아 설치 비용이 많이 들고, 변경 등의 관리 작업도 어려운 문제점이 있었다. 또한, 애플리케이션 인스턴스를 개별적으로 띄워서 관리하며 관련 데이터베이스도 사용 기관에 거의 일대일로 제공해야하므로 규모의 경제를 실현하기 어려운 문제점이 있었다. SaaS 방식의 애플리케이션 서비스는 사용자가 필요한 소프트웨어를 온라인 서비스로 이용할 수 있도록 하는 최신의 소프트웨어 배포 모델로 정의되며, 사용자가 온라인을 통해 소프트웨어를 사용하고 그에 대한 비용만 지불하는 방식으로 복잡한 소프트웨어 및 하드웨어의 관리라는 부담에서 벗어날 수 있다[2]. 또한 SaaS 서비스는 성숙도 level에 따라 하나의 애플리케이션으로 다중테넌트를 지원하도록 테넌트별 설정 기능을 제공하며, level 3부터는 하나의 애플리케이션 인스턴스로 다수의 기관을 동시에 지원할 수 있는 특징을 가진다[2,7]. 이렇게 다수의 테넌트들을 하나의 애플리케이션 인스턴스로 지원하고 각 테넌트에 맞게 개별화를 용이하게 함으로써 수요자에게 커스터마이즈에 많은 비용이 들지 않고, 제공 측면에서도 서비스 제공에 저비용이 소요되어 수익을 향상시키는 규모의

경제를 실현할 수 있다. 본 연구에서는 SaaS 플랫폼에서 제공하는 그러한 설정 기능들을 분류하고, 설정 기능 중 데이터 스키마 설정 기능을 지원하기 위한 애플리케이션 구현 방식중 하나를 들어 설명하고자한다. 본 설정 기능들은 온라인으로 제공하는 애플리케이션을 모바일 단말에 맞게 커스터마이즈하기 위해서도 사용 가능하다.

2. SaaS 플랫폼에서 설정 기능 관련 연구 동향

SaaS 방식의 서비스는 사용자가 필요한 애플리케이션을 온라인 서비스로 이용하고 그에 대한 비용만 지불하면 되는 방식으로 복잡한 소프트웨어 및 하드웨어에 대한 소요 비용 및 관리에 대한 부담에서 벗어날 수 있는 장점을 가지고 있는 소프트웨어 서비스 방식이어서 소프트웨어 실행, 서비스 제공 방식의 새로운 패러다임으로 대두되고 있다. SaaS 플랫폼에서 지원하는 애플리케이션의 대표적은 장점으로 다수의 테넌트들을 하나의 애플리케이션 인스턴스로 지원하고 각 테넌트에 맞게 설정하여 개별화를 가능하게 함으로써, 커스터마이즈에 많은 비용이 들지 않는 점을 들 수 있다. 여기서 테넌트(Tenant)는 여러 사용자 단말이 속한 하나의 사용자 집단으로 회사, 기관, 단체 등을 의미한다. 즉, 테넌트는 복수의 사용자 단말을 이용하여 애플리케이션을 사용하는 하나의 사용자 집단이다. 여기서 테넌트 관리자는 테넌트의 온라인 애플리케이션 이용 환경을 설정하고 관리한다. 사용자는 인터넷을 통해 SaaS 플랫폼 기반 온라인 애플리케이션에 접속하여 테넌트 관리자에 의해 설정된 온라인 애플리케이션 이용 환경에 따라 애플리케이션을 이용한다. 이와 같이 테넌트별 커스터마이즈가 가능하게 지원하는 SaaS 플랫폼의 설정 기능 관련 연구 동향을 살펴보면 아래와 같이 Salesforce.com의 Force.com과 한국전자통신연구원의 SaaSpia™ 의 경우로 정리할 수 있다.

2.1 해외 관련 연구 동향

SaaS 기반 플랫폼 관련 선두업체들 중의 하나인 Salesforce.com은 CRM 솔루션으로 시작하여 지금은 SaaS 플랫폼과 클라우드 플랫폼 기술을 개발하여 새로운 시장을 공략하고 있다.

Salesforce.com의 Force.com 플랫폼은 ‘Global, Trusted, Secure Infrastructure’ 스택, ‘Database as a Service’ 스택, ‘Integration as a Service’ 스택, ‘Logic as a Service’ 스택, ‘User Interface as a Service’ 스택, 그리고 ‘Application Exchange’ 스택으로 구성되어 있다. 이 스택들 중 ‘Logic as a Service’ 스택은 고객사의 비즈니스 프로세스와 요구사항에 맞게 워크플로를 쉽게 생성할 수 있도록 공통적이며 재사용 가능한 컴포넌트들을 활용할 수 있도록 지원한다. 유연성을 더 높이기 위해서는 Apex라는 온디맨드 프로그래밍 언어를 사용하여 가상적으로 비즈니스 로직과 기능을 포함시킬 수 있다. 또한, Force.com은 플랫폼상에서 실행되는 애플리케이션의 사용자 인터페이스를 새로 만들고 커스터마이즈가 가능하도록 지원하기 위해 ‘User Interface as a Service’ 스택에서 두 가지 방법을 제공한다. 애플리케이션에 온라인 설정 공간을 통해 간단히 드래그앤드롭(drag-and-drop) 인터페이스를 사용하여 페이지의 레이아웃이나 데이터 필드를 변경하거나 다른 사용자를 위해 다른 뷰(view)를 생성할 수 있다. 다른 방법으로는 Force.com에 포함된 ‘Visualforce’ 라는 UI 생성 도구를 이용해서 개발자가 사용자 인터페이스를 생성할 수 있다[3,4].

2.2 국내 관련 연구 동향

국내에서 개발중인 SaaS 기반 서비스를 지원하는 플랫폼 중 대표적인 예로 한국전자통신연구원에서 개발중인 SaaSpia™을 들 수 있다. SaaSpia™는 SaaS 성숙도 Level 3를 목표로 개발되고 있어서 SaaSpia™ 기반으로 애플리케이션을 개발했을 때 다중테넌트를 지원하여 테넌트 별로 개별화할 수 있는 설정 기능을 제공한다. 설정 기능으로는 사용자 인터페이스(UI) 설정 기능, 데이터 스키마 설정 기능, 비즈니스 로직 설정 기능, 그리고 워크플로우 설정 기능을 지원하는 것을 목표로 하고 있다[5,6].

3. SaaS 플랫폼에서 제공하는 설정 기능 분류

위에서와 같이 SaaS 플랫폼에서 대표적으로 지원하는 여러 측면의 설정 기능을 아래와 같이 분류할 수 있다.

3.1 사용자 인터페이스(UI) 설정 기능

일반적으로 UI 설정 기능을 지원하기 위해서는 웹페이지 개발시부터 컴포넌트 단위로 개발되어 있고, 테넌트 관리자가 설정시 설정 정보가 저장된 후 테넌트에 속하는 사용자에게 UI가 로딩될 때 테넌트별 설정정보에 따라 컴포넌트화된 UI 요소가 랜더링되어 제시된다. 동일한 애플리케이션도 테넌트 별로 다른 UI를 제공하여 웹페이지를 맞춤형으로 설정 가능하게 지원한다. UI 생성이나 설정을 위해서는 플랫폼에서 UI 개발과 설정을 위한 별도의 도구가 사용될 수 있다.

3.2 데이터 스키마 설정 기능

애플리케이션이 개발될 때 기본적으로 제공되는 데이터 스키마 외에 테넌트 별로 관리자가 필요에 따라 확장이 가능한 스키마를 제공하여 테넌트 업무 환경에 맞도록 스키마를 설정하여 사용할 수 있도록 지원한다. 이 데이터 스키마 설정 기능은 애플리케이션에서 확장된 데이터 스키마가 비즈니스 로직과도 관련이 되기 때문에 내부적으로 비즈니스 로직 설정 기능과도 연관된다.

3.3 비즈니스 로직 설정 기능

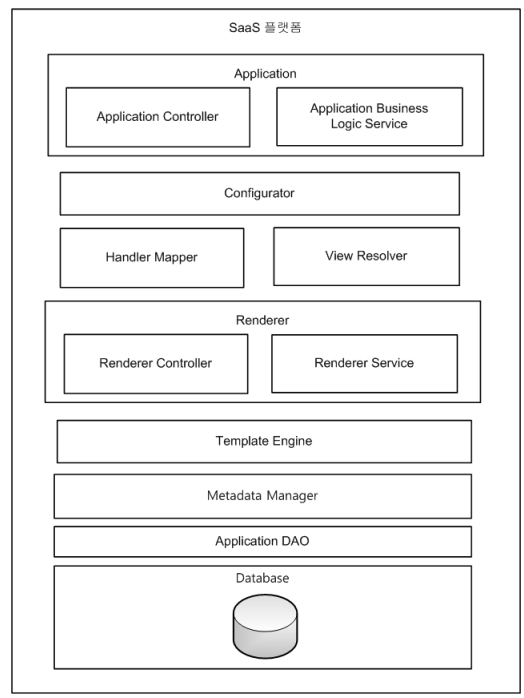
웹기반 애플리케이션 개발자가 개발한 비즈니스 로직을 테넌트별로 설정하여 맞춤화할 수 있는 기능을 지원한다. 일반적으로 웹 애플리케이션 사용자는 개발자가 제공하는 비즈니스 로직의 범위 내에서 기능을 동작시킬 수 있으나 비즈니스 로직 설정 기능을 이용해서 테넌트 관리자가 테넌트에 맞게 비즈니스 로직까지 변경하여 애플리케이션을 커스터마이즈하여 사용할 수 있다. 비즈니스 로직 설정 기능은 별도의 온디맨드 프로그래밍 언어를 지원하여 테넌트 관리자가 비즈니스 로직을 변경 가능하게 지원하거나, 자바 스크립트를 이용한 방법과 자바 컨트롤러를 이용한 방법 등으로 구분될 수 있다.

3.4 워크플로우 설정 기능

웹기반 애플리케이션 이용 시 테넌트 별로 워크플로우를 다르게 설정하여 테넌트 별로 설정한 프로세스에 따라 업무를 수행할 수 있도록 지원하는 기능이다. 테넌트 관리자가 설정한 테넌트 사용의 역할이나 권한에 따라 프로세스가 진행되며, 테넌트 사용자에게 테넌트별로 다르게 설정된 프로세스 순서대로 흘러가는 UI에 따라 업무를 수행한다.

4. 확장 필드를 응용한 데이터 스키마 설정 기능 연구

본 연구에서는 SaaS 플랫폼이 테넌트에게 제공하는 설정 기능들 중 데이터 스키마 설정 기능 제공 방법에 대해 예를들어 설명하고자 한다. 확장 필드를 응용한 데이터 스키마 설정 기능을 제공하기 위해서 SaaS 플랫폼은 (그림 1)과 같은 세부 모듈들을 포함할 수 있다. 아래의 UI 예((그림 2), (그림 3))와 같이 개발자가 기본적으로 개발한 애플리케이션의 데이터 목록 UI에 테넌트 관리자가 추가적으로 데이터 필드를 확장하여 테넌트별로 개별의 애플리케이션처럼 이용할 수 있는 확장성과 유연성을 가진다. 아래의 화면 예에서 '담당자 전화번호' 필드는 테넌트 관리자가 해당 테넌트의 애플리케이션 사용에 필요하다고 판단하여 확장 필드를 이용하여 설정된 부분이다. 위의 UI 예와 같이 지원하기 위해, <표 1>의 테이블과



(그림 1) SaaS 플랫폼 구조도 예

No.	회사명	사업자등록번호	담당자이름
1	만도기계	6734264391	이수정
2	인도 회사	4028154166	Andora
3	일성산업	4028154166	김성진
4	베트남회사	4028154166	트란 트롱
5	Henry Corp	12345	Alice
6	James & Jane Company	17890	Christina
7	Percy	40264	Antonio
8	Emily	40265	Allen Doruean
9	Thomas	40267	Forcy Decran
10	Gorden	40268	Murcan Stephan
11	Donald	56789	Tepance

(그림 2) 기본 list UI 예

No.	회사명	사업자등록번호	담당자이름	담당자전화번호
1	만도기계	6734264391	이수정	02-323-3433
2	인도 회사	4028154166	Andora	235-5682
3	일성산업	4028154166	김성진	042-658-7852
4	베트남회사	4028154166	트란 트롱	02-839-2393
5	Henry Corp	12345	Alice	02-839-2393
6	James & Jane Company	17890	Christina	090-298-2738
7	Percy	40264	Antonio	032-9375-7585
8	Emily	40265	Allen Doruean	794-3419-1341
9	Thomas	40267	Forcy Decran	034-1353-1353
10	Gorden	40268	Murcan Stephan	034-135-1341
11	Donald	56789	Tepance	03-1535-1345

(그림 3) 확장 필드를 추가한 list UI 예

〈표 1〉 확장 필드 지원을 위한 DB 스키마 구성 예

COMPANY	
COMPANY_ID	INT
COMPANY_NAME	VARCHAR
BIZ_REGIST_NUM	VARCHAR
PERSON_IN_CHARGE_NAME	VARCHAR

[기본 테이블]

TENANT_ID	INT
C_FLEX1	VARCHAR
C_FLEX2	VARCHAR
C_FLEX3	VARCHAR
C_FLEX4	VARCHAR

[확장용 테이블]

같은 기본 테이블과 함께 확장 필드를 테넌트별로 매핑한 정보와 테넌트별로 설정한 정보를 담는 테이블 등으로 DB 스키마를 구성할 수 있다. 확장 필드에는 테넌트별로 다양한 형태의 정보가 담길 수 있고, 다른 테이블에 그 정보의 형태를 저장한다.

본 예의 설명을 위해 SaaS 플랫폼에서 웹기반의 가용한 각 UI 컴포넌트들은 메타데이터화 되어 이미 데이터베이스에 저장되어 있다고 가정한다. SaaS 플랫폼 상에서 애플리케이션 개발자가 이미 저장된 UI 컴포넌트를 이용하여 애플리케이션을 개발하며, 확장 필드를 고려하지 않고 고정 필드만 고려하여 애플리케이션을 개발한다. 그리고 테넌트 관리자가 위의 구조도(그림 1)에서 Configurator(설정 도구)를 이용하여 해당 애플리케이션을 설정시에 확장 필드를 선택하고 설정하여 테넌트별로 추가적인 필드를 사용할 수 있게 한다. 사용자는 고정 필드, 확장 필드와 무관하게 UI에 제시된 데이터를 이용하여 애플리케이션을 사용하면 된다.

4.1 확장 필드를 포함한 UI 랜더링 과정

먼저 개발 및 설정이 완료된 애플리케이션을 이용시에, 확장필드(Flexible field)를 포함한 웹기반 UI를 랜더링하는 과정을 설명하기로 한다. 사용자 PC의 웹브라우저를 통해 특정 테넌트의 특정 페이지 URL이 요청되면, 위의 구조도(그림 1)의 모듈들 중 Handler Mapper가 그 URL 요청에 해당하는 Handler(Controller)를 매핑해준다. 특정 테넌트의 특정 페이지 URL이 요청될 때 테넌트 ID, 페이지 ID가 파라미터로 계속 전달된다. 해당 Renderer controller 는 Renderer Service 모듈에 해당 페이지 구성을 위한 Javascript 코드 생성을 요청한다. 그에 따라, Renderer Service 모듈은 Metadata Manager에 해당 페이지에 필요한 UI 컴포넌트를 요청한다. 이에, Metadata Manager는 Database에서 알맞은 data를 가져오기 위한 SQL 문을 동적으로 생성하여 데이터를 요청한다. 다음으로, Database는 <표 2>의 예와 같은 UI 메타 데이터를 Metadata Manager에 반환하고, 그 데이터를 받은 Metadata Manager는 관계형(Relational) 자료를 객체(Object)형 자료로 변환하여 UI 컴포넌트 모델 객체를 생성하여 Renderer Service 모듈에 반환한다.

이에 따라, Renderer Service 모듈은 Template Engine에 해당 UI에 관한 Template Code와 Hash Map과 함께 UI 컴포넌트 모델 객체를 전달하여

〈표 2〉 해당 페이지에 관한 UI 메타 데이터 예

CompID	CompType	AttrID	AttrName	StoreURL	Label
1	GridPanel	null	null	/getProductList.do	제품목록
2	GridColumn	1	PRODUCT_NAME	null	제품명
3	GridColumn	2	C_FLEX1	null	확장필드

UI를 만들기 위한 Javascript 코드 생성을 요청한다. 각 UI 컴포넌트별로 Template Code가 미리 정의되어 있어야한다. Template Code에서 변하는 부분은 <표 3>의 예와 같이 변수로 표시된다. Template Engine은 <표 4>의 예(Grid 형 UI)와 같은, UI에 해당하는 Javascript 코드를 생성해서 Renderer Service 모듈에 반환한다.

그리고, Renderer Service 모듈은 반환 받은 Javascript 코드를 Renderer Controller에 전달한다. Renderer Controller에 전달시에 향후 어떤 View를 사용할 것인지 파라미터로 지정되어 있어야한다. Renderer Controller는 그 Javascrip 코드를 이용해 ModelandView 객체를 생성해서 View Resolver에 반환한다. View Resolver는 해당 UI에 맞는 View (예) productui1.jsp)를 이용해

〈표 3〉 Template Code 및 Hash Map 예

[Template Code]

```
GridStore = function(){

GridStore.superclass.constructor.call(this, {
            remoteSort: true,
            proxy: new Ext.data.HttpProxy({
              url: '$storeUrl'
            }),
            reader: new Ext.data.JsonReader({
            [{
              name: '$dataIndex',
              mapping: '$label'
            }
            ])
        });
};
```

[Hash Map]

```
storeUrl : '/getProductList.html'
{
dataIndex[0]: 'PRODUCT_NAME'
label[0]: '제품명'
dataIndex[1]: 'C_FLEX1'
label[1]: '확장필드'
}
```

〈표 4〉 생성된 Javascript 코드 예

```
GridStore = function(){
        GridStore.superclass.constructor.call(this, {
        remoteSort: true,
        proxy: new Ext.data.HttpProxy({
            url: '/getProductList.html'
        }),
        reader: new Ext.data.JsonReader({
        [{
            name: 'PRODUCT_NAME',
            mapping: '제품명'
        },
        {
            name: 'C_FLEX1',
            mapping: '확장필드'
        }
        ])
    });
};
```

HTML 페이지를 생성하여 사용자 PC의 웹브라우저에 전달한다.

4.2 UI 랜더링후 데이터 로딩 과정

이전 과정에서 생성된 HTML 페이지를 웹브라우저가 해석해서 로딩하는 중에 필요한 실제 데이터를 가져오는 단계는 Ajax와 같이 웹브라우저에서 동적 UI가 구동될 시에 자동으로 데이터를 로딩을 요청하는 경우와 유사하다. 단, 앞에서 설명한 UI 랜더링 경우와는 다르게, 실제 데이터는 애플리케이션과 관련되므로 App. Controller에 URL을 파라미터로(예) getProductList.html) 요청한다. 이 URL를 파라미터로 요청하면 그 URL에 해당하는 데이터가 JSON 이나 XML 코드 형태로 반환되는데 이 형태의 데이터는 위해서 생성된 Javascript 형태로 해석된다. 먼저, 사용자 PC의 웹브라우저는 App. Controller에 URL에 해당하는 데이터를 요청한다. App. Controller는 App. Business Logic Service 모듈에 해당 메서드를 호출하여 데이터를 요청한다. App. Business Logic

〈표 5〉 Metadata Manager API 호출 예

```
Function Name
    - Select()

Params
    - TenantId : Integer : in
    - TableName : String : in
    - FieldName : String : in
    - Fixed Field : Model : out
    - Flexible Field : JSON : out
```

Service 모듈은 App. DAO를 통해 맞는 Metadata Manager의 API를 호출한다. API 호출에는 <표 5>의 예와 같이, 필요한 필드명 등이 파라미터로 포함된다. Metadata Manager는 요청한 데이터를 가져올 SQL문을 동적으로 생성하여 데이터베이스에 필요한 데이터를 요청한다.

이 데이터 요청에 대해 데이터베이스는 <표 6>의 예와 같이 확장 필드를 포함한 필요 데이터를 추출하여 Metadata Manager에 반환한다.

이에 따라, 위의 API 파라미터에 맞게 Metadata Manager는 고정 필드의 관계형(Relational) 데이터는 모델 객체로 생성하고 확장 필드 데이터는 JSON 형태나 XML 코드 형태로 생성해서 반환한다.

App. DAO는 반환된 고정 필드 모델 객체들과 확장 필드 JSON 형태나 XML 코드 형태의 데이터를 App. Business Logic Service 모듈에 반환된다. 반환된 고정 필드 모델 객체의 데이터는 App. 로직에서 연산 등을 거친 후에 결과 값과 JSON 형태의 확장 필드의 데이터를 <표 9>의 예와 같이, JSON 형태나 XML 코드 형태로 통합한다. 앞에서 제시되었던 Javascript 형태로 해석되기 위해서이다. 통합된 코드는 App. Controller에 전달되고, 다시 사용자 PC의 웹브라우저에 전달되어 Javascript에 해당 데이터들이 채워져서 이전의 UI와 함께 실제 데이터까지 로딩이 완료된다.

〈표 6〉 확장 필드를 포함하여 추출된 데이터 예

PRODUCT_NAME	C_FLEX1	TENANT_ID
연필	연필 설명 (확장필드)	1
볼펜	볼펜 설명 (확장필드)	1
지우개	지우개 설명 (확장필드)	1
형광펜	형광펜 설명 (확장필드)	1

〈표 7〉 고정 필드 데이터 부분 예

PRODUCT_NAME
연필
볼펜
지우개
형광펜

〈표 8〉 확장 필드 데이터 부분 JSON 코드 예

```
{
        C_FLEX1:
        {
            '연필 설명 (확장필드)',
            '볼펜 설명 (확장필드)',
            '지우개 설명 (확장필드)',
            '형광펜 설명 (확장필드)'
        }
}
```

〈표 9〉 통합된 JSON 코드 예

```
[
    {
        PRODUCT_NAME : '연필',
    },
    {
        PRODUCT_NAME : '볼펜',
    },
    {
        PRODUCT_NAME : '지우개',
    },
    {
        PRODUCT_NAME : '형광펜',
     },
        C_FLEX1:
        {
            '연필 설명 (확장필드)',
            '볼펜 설명 (확장필드)',
            '지우개 설명 (확장필드)',
            '형광펜 설명 (확장필드)'
}]
```

4.3 확장 필드를 포함한 데이터가 처리되고 저장이나 업데이트되는 과정

확장 필드를 포함한 데이터가 처리되고 저장이나 업데이트되는 과정은 내부적으로 JSON이나 XML 코드 형태로 된 데이터를 이용하여 로직을 거친 후 저장이나 업데이트를 하기 위해, 먼저, 웹브라우저에서 사용자에 의해 요청되어 App. Controller에 요청이 전달된다. App. Controller는 해당 로직을 호출하여 요청하고, App. Business Logic Service 모듈은 위에서와 같은 코드 형태의 데이터를 분리해서 고정 필드 데이터 모델 객체를 생성한다. 그 고정 필드 데이터를 이용하여 애플리케이션에서 내부 로직이 처리되고, 저장이나 업데이트를 위해 App. DAO로 확장 필드의 JSON 코드 및 고정 필드 데이터 모델 객체가 전달되면서 데이터 처리가 요청된다. App. DAO는 이 작업에 해당하는 Metadata Manager API를 호출한다.

이에 따라 Metadata Manager는 확장 필드의 코드 형태 데이터와 고정 필드 데이터 모델 객체 등의 입력된 값을 이용하여 데이터 모델을 생성한다. 그리고, 데이터베이스에 저장이나 업데이트를 위한 SQL문이 동적으로 생성되고 데이터베이스에 요청되어 실행된다.

〈표 10〉 데이터 업데이트를 위한 Metadata Manager API 호출 예

```
Function Name
    - Update()
Params
    - TenantId : Integer : in
    - TableName : String : in
    - FieldName : String : in
    - Fixed Field : Model : in
    - Flexible Field : JSON : in
```

5. 결 론

본 연구에서 제시하는 SaaS 플랫폼 기반의 애플리케이션의 데이터 테이블은 고정 필드와 확장 필드로 구성된다. 고정 필드는 모든 테넌트들에게 공통적으로 서비스를 제공하기 위한 데이터 필드로 애플리케이션 개발시 개발자에 의해 설계되고 구현되며, 확장 필드는 테넌트 관리자에 의해 설정이 가능하도록 지원하는 데이터 필드로서 동일한 확장 필드라고 해도 테넌트별로 다른 기능으로 활용될 수 있다. 즉 하나의 확장 필드는 테넌트에 따라 다른 필드 타입을 정의할 수 있기 때문에 테넌트별로 서로 다른 용도로 사용할 수 있다. 본 논문을 통해 기존의 애플리케이션의 고정된 데이터 필드 외의 확장 필드와 메타 데이터를 이용하여 다수의 테넌트에 대해 각각의 개별화된 온라인 애플리케이션을 제공하는 데이터 스키마 설정 기능 제공 방법을 제시하였다. 본 연구에서 설명된 데이터 스키마 설정 기능 제공 방법은 메타 데이터를 이용하여 하나의 애플리케이션을 각 테넌트에 개별화된 애플리케이션으로 변경하여 제공함으로써, 다수의 테넌트들을 하나의 애플리케이션 인스턴스(Single-instance, Multi-tenant)로 지원할 수 있다. 제시된 SaaS 플랫폼 기반의 온라인 애플리케이션 제공 방법은 다수의 테넌트들을 하나의 애플리케이션 인스턴스로 지원하고 각 테넌트에 맞게 개별화를 용이하게 함으로써, 사용자 측면에서 커스터마이즈에 많은 비용이 들지 않고, 애플리케이션 인스턴스를 테넌트마다 개별적으로 띄우지 않고 하나의 애플리케이션 인스턴스로 다수의 테넌트 지원이 가능하며 데이터베이스 자원을 공유하므로 저비용이 소요되어 서비스 제공자 측면에서도 제공 비용을 낮춰 수익을 향상시키는 규모의 경제를 실현할 수 있다. 또한 본 논문에서 분류한 설정 기능들을 응용하면

모바일 단말을 이용하는 사용자들의 환경에 커스터마이즈할 수 있는 가변적 UI나 데이터 필드를 구성하여 지원할 수 있다.

참 고 문 헌

[1] 최형광, 클라우드 컴퓨팅과 비즈니스의 진화, 디지털데일리, 2009년 5월.

[2] 김형환 외 12, SaaS 기술 개발 동향, 전자통신동향분석, 24권 4호, pp.15-17, 2009년 8월.

[3] Force.com: A Comprehensive Look at the World's Premier Cloud-Computing Platform, pp.8-9, http://wiki.developerforce.com/index.php/Documentation, 2009.

[4] 김형환 외 12, SaaS 기술 개발 동향, 전자통신동향분석, pp.19-20, 2009년 8월.

[5] 박경현 외 2, 다중 테넌트 지원을 위한 SaaS 어플리케이션 설정 환경, 한국인터넷정보학회 추계학술발표대회 논문집, 제11권2호, pp.251-252, 2010년.

[6] SaaSpia™ 홈페이지, www.saaspia.org

[7] 민병원 외 1, SaaS 기반 멀티테넌트 환경을 지원하는 통합전자도서관시스템 구현, 한국콘텐츠학회논문지, v.11 No.5, 2011년.

[8] 이지현 외 10, SaaS 플랫폼 기술 및 개발 동향, 전자통신동향분석, 26권 5호, 2011년 10월.

저 자 약 력

오 병 택

이메일 : btoh@etri.re.kr

- 1999년 전주대학교 산업공학과(학사)
- 2001년 아주대학교 산업공학과(석사, HCI 전공)
- 2001년~현재 한국전자통신연구원 선임연구원
- 관심분야 : 온라인 SW 플랫폼, 클라우드컴퓨팅, P2P 네트워킹, UX 분야

원 희 선

이메일 : hswon@etri.re.kr

- 1990년 연세대학교 전산과학과(학사)
- 1992년 KAIST 전산학과(석사, 데이터베이스 전공)
- 1992년~1999년 한국방송 기술연구소 연구원
- 2000년~현재 한국전자통신연구원 선임연구원
- 관심분야 : 온라인 SW 플랫폼, 데이터 마이닝

허 성 진

이메일 : sjheo@etri.re.kr

- 1990년 경북대학교 전자공학과(학사)
- 1992년 경북대학교 컴퓨터공학과(석사)
- 1999년 경북대학교 컴퓨터공학과(박사)
- 1999년~2001년 창신대학 인터넷정보과 전임강사
- 2001년~현재 한국전자통신연구원 책임연구원
- 2009년~현재 한국전자통신연구원 SW서비스연구팀장
- 관심분야 : 온라인 SW 플랫폼, 클라우드컴퓨팅