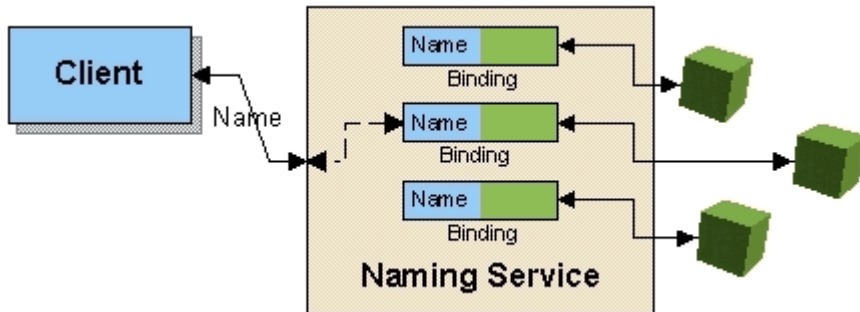


개요

Naming 서비스는 Java Naming and Directory Interface(JNDI) API를 이용하여 자원(Resource)를 찾을 수 있도록 도와주는 서비스이다. Naming 서비스를 지원하는 Naming 서버에 자원을 등록하여 다른 어플리케이션에서 사용할 수 있도록 공개하고, Naming 서버에 등록되어 있는 자원을 찾아와서 이용할 수 있게 한다.

- Naming 서비스
 - 개요
 - 주요 개념
- 설명
 - Spring XML Configuration 설정
 - JndiTemplate 클래스 사용
- 참고자료



주요 개념

Java Naming and Directory Interface(JNDI)

Java Naming and Directory Interface(JNDI)는 Java 소프트웨어 클라이언트가 이름(name)을 이용하여 데이터 및 객체를 찾을 수 있도록 도와주는 디렉토리 서비스에 대한 Java API이다.

- 참조 : <http://en.wikipedia.org/wiki/JNDI>

설명

Naming 서비스는 사용하는 방식에는 Spring XML Configuration 파일에 설정하는 방식과 JNDI API를 wrapping한 JndiTemplate class를 사용하는 방식이 있다.

- **Spring XML Configuration 파일에 설정하는 방식 :**
Spring XML Configuration 설정파일에 JNDI 객체를 bean으로 등록하는 방식으로, JNDI 객체를 Lookup만 할 수 있다. 일반적으로 가장 많이 사용된다.
- **JNDI API를 wrapping한 JndiTemplate class를 사용하는 방식 :**
Spring Framework에서 JNDI API를 쉽게 사용할 수 있도록 제공하는 JndiTemplate class를 직접 사용하는 방식으로, JNDI API 기능을 모두 사용해야 할 경우 사용하는 방식이다.

Spring XML Configuration 설정

Spring Framework는 XML Configuration 파일에 JNDI 객체를 설정할 수 있다. 단, 설정 파일을 통해서 JNDI 객체를 lookup하는 것만 가능하므로, bind, rebind, unbind 기능을 사용하려면 [Using JndiTemplate](#) 방식을 사용해야 한다.

Spring Framework은 XML Configuration을 간편하게 할 수 있게 하기 위해 2.0 버전부터 jee tag를 제공하고 있다. 전자정부 표준프레임워크는 Spring 2.5 이상을 기반으로 하기 때문에 본 가이드는 jee tag를 사용한 방식만을 설명한다.

설정

jee tag를 사용하기 위해서는 Spring XML Configuration 파일의 머릿말에 namespace와 schemaLocation를 추가해야 한다.

- namespace : xmlns:jee="http://www.springframework.org/schema/jee"
- schemaLocation : http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-2.5.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jee="http://www.springframework.org/schema/jee"
       xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-2.5.xsd">

    <!-- <bean/> definitions here -->

</beans>
```

jndi-lookup tag

jndi-lookup tag는 JNDI 객체를 찾아서 bean으로 등록해주는 tag이다.

tag 설명

```
<jee:jndi-lookup id="bean id"
                 jndi-name="jndi name"
                 cache="true or false"
                 resource-ref="true or false"
                 lookup-on-startup="true or false"
                 expected-type="java class"
                 proxy-interface="java class">

    <jee:environment>
        name=value
        ping=pong
        ...
    </jee:environment>
</jee:jndi-lookup>
```

jndi-lookup tag는 Spring Framework의 JndiObjectFactoryBean class와 1:1로 매핑된다. tag의 attribute 값은 다음과 같다.

Attribute	설명	Optional	Data Type	Default 값	비고
id	Spring XML Configuration의 bean id이다.	N	String		
jndi-name	찾고자 하는 JNDI 객체의 이름이다.	N	String		
cache	한번 찾은 JNDI 객체에 대한 cache여부를 나타낸다.	Y	boolean	true	
resource-ref	J2EE Container 내에서 찾을지 여부를 나타낸다.	Y	boolean	false	
lookup-on-startup	시작시에 lookup을 수행할지 여부를 나타낸다.	Y	boolean	true	
expected-type	찾는 JNDI 객체를 assign할 타입을 나타낸다.	Y	Class		값이 지정되지 않았을 경우 무시한다.
proxy-interface	JNDI 객체를 사용하기 위한 Proxy Interface이다.	Y	Class		값이 지정되지 않았을 경우 무시한다.

jndi-lookup tag의 element인 environment tag는 JNDI Environment 변수값을 등록할 때 사용한다. environment tag는 'foo=bar' 와 같이 <변수명>=<변수값> 형태의 List를 값으로 가진다.

예제

■ Simple

가장 단순한 설정으로 이름만을 사용하여 JNDI 객체를 찾아준다. 아래 이름 "jdbc/MyDataSource"로 등록 있는 JNDI 객체를 찾아 "userDao" Bean의 "dataSource" property로 Dependency Injection하는 예제이다.

```
<jee:jndi-lookup id="dataSource" jndi-name="jdbc/MyDataSource" />

<bean id="userDao" class="com.foo.JdbcUserDao">
    <!-- Spring will do the cast automatically (as usual) -->
    <property name="dataSource" ref="dataSource" />
</bean>
```

■ With single JNDI environment settings

아래는 단일 JNDI 환경 설정을 사용하여 JNDI 객체를 찾아오는 예제이다.

```
<jee:jndi-lookup id="dataSource" jndi-name="jdbc/MyDataSource" >
    <jee:environment>foo=bar</jee:environment>
</jee:jndi-lookup>
```

- With multiple JNDI environment settings

아래는 복수 JNDI 환경 설정을 사용하여 JNDI 객체를 찾아오는 예제이다.

```
<jee:jndi-lookup id="dataSource" jndi-name="jdbc/MyDataSource" >
    <!-- newline-separated, key-value pairs for the environment (standard Properties format) -->
    <jee:environment>
        foo=bar
        ping=pong
    </jee:environment>
</jee:jndi-lookup>
```

- Complex

아래는 이름 외 다양한 설정을 통해 JNDI 객체를 찾아오는 예제이다.

```
<jee:jndi-lookup id="dataSource"
    jndi-name="jdbc/MyDataSource"
    cache="true"
    resource-ref="true"
    lookup-on-startup="false"
    expected-type="com.myapp.DefaultFoo"
    proxy-interface="com.myapp.Foo" />
```

local-slsb tag

local-slsb tag는 EJB Stateless SessionBean을 참조하기 위한 tag이다.

tag 설명

```
<jee:local-slsb id="bean id"
    jndi-name="JNDI name"
    business-interface="Java Class"
    cache-home="true or false"
    lookup-home-on-startup="true or false"
    resource-ref="true or false">
    <jee:environment>
        name=value
        ping=pong
        ...
    </jee:environment>
</jee:local-slsb>
```

local-slsb tag는 Spring Framework의 LocalStatelessSessionProxyFactoryBean class와 1:1로 매핑된다. tag의 attribute는 다음과 같다.

Attribute	설명	Optional	Data Type	Default 값	비고
id	Spring XML Configuration의 bean id이다.	N	String		
jndi-name	찾고자 하는 EJB의 JNDI 이름이다.	N	String		
business-interface	Proxing할 EJB의 Business interface이다.	N	Class		
cache-home	한번 찾은 EJB Home 객체에 대한 cache여부를 나타낸다.	Y	boolean	true	
lookup-home-on-startup	시작 시에 lookup을 수행할지 여부를 나타낸다.	Y	boolean	true	

resource-ref	J2EE Container 내에서 찾을지 여부를 나타낸다.	Y	boolean	false	
--------------	----------------------------------	---	---------	-------	--

local-slsb tag의 element인 environment tag는 JNDI Environment 변수값을 등록할 때 사용한다. environment tag는 'foo=bar' 와 같이 <변수명>=<변수값> 형태의 List를 값으로 가진다.

예제

■ Simple

간단히 사용하는 예제이다.

```
<jee:local-slsb id="simpleSlsb" jndi-name="ejb/RentalServiceBean"
    business-interface="com.foo.service.RentalService" />
```

■ Complex

local-slsb tag를 사용하기 위해 다양한 설정 값을 이용하는 예제이다.

```
<jee:local-slsb id="complexLocalEjb"
    jndi-name="ejb/RentalServiceBean"
    business-interface="com.foo.service.RentalService"
    cache-home="true"
    lookup-home-on-startup="true"
    resource-ref="true" />
```

remote-slsb tag

remote-slsb tag는 remote EJB Stateless SessionBean을 참조하기 위한 tag이다.

tag 설명

```
<jee:remote-slsb id="bean id"
    jndi-name="JNDI name"
    business-interface="Java Class"
    cache-home="true or false"
    lookup-home-on-startup="true or false"
    resource-ref="true or false"
    home-interface="Java Class"
    refresh-home-on-connect-failure="true or false">
    <jee:environment>
        name=value
        ping=pong
        ...
    </jee:environment>
</jee:remote-slsb>
```

remote-slsb tag는 Spring Framework의 SimpleRemoteStatelessSessionProxyFactoryBean class와 1:1로 매핑된다. tag의 attribute는 아래와 같다.

Attribute	설명	Optional	Data Type	Default 값	비고
id	Spring XML Configuration의 bean id이다.	N	String		
jndi-name	찾고자 하는 EJB의 JNDI 이름이다.	N	String		
business-interface	Proxing할 EJB의 Business interface이다.	N	Class		
cache-home	한번 찾은 EJB Home 객체에 대한 cache여부를 나타낸다.	Y	boolean	true	
lookup-home-on-startup	시작 시에 lookup을 수행할지 여부를 나타낸다.	Y	boolean	true	
resource-ref	J2EE Container 내에서 찾을지 여부를 나타낸다.	Y	boolean	false	
home-interface	EJB Home interface이다.	Y	Class		
refresh-home-on-connect-failure	연결 실패 시, EJB Home을 refresh할지 여부를 나타낸다.	Y	boolean	false	

remote-slsb tag의 element인 environment tag는 JNDI Environment 변수값을 등록할 때 사용한다. environment tag는 'foo=bar' 와 같이 <변수명>=<변수값> 형태의 List를 값으로 가진다.

예제

■ Simple

간단히 사용하는 예제이다.

```
<jee:remote-slsb id="complexRemoteEjb"
    jndi-name="ejb/MyRemoteBean"
    business-interface="com.foo.service.RentalService"
    home-interface="com.foo.service.RentalService" />
```

■ Complex

remove-slsb tag를 사용하기 위해 다양한 설정 값을 이용하는 예제이다.

```
<jee:remote-slsb id="complexRemoteEjb"
    jndi-name="ejb/MyRemoteBean"
    business-interface="com.foo.service.RentalService"
    cache-home="true"
    lookup-home-on-startup="true"
    resource-ref="true"
    home-interface="com.foo.service.RentalService"
    refresh-home-on-connect-failure="true" />
```

JndiTemplate 클래스 사용

JndiTemplate class는 JNDI API를 쉽게 사용할 수 있도록 제공하는 wrapper class이다.

bind

아래 JndiTemplateSample class의 bind 메소드는 JndiTemplate을 이용하여 argument 'resource'를 argument 'name'으로 JNDI 객체로 bind한다.

```
import javax.naming.NamingException;
import org.springframework.jndi.JndiTemplate;
...
public class JndiTemplateSample
{
    private JndiTemplate jndiTemplate = new JndiTemplate();
    ...
    public boolean bind(final String name, Object resource)
    {
        try
        {
            jndiTemplate.bind(name, resource);
            return true;
        }
        catch (NamingException e)
        {
            e.printStackTrace();
            return false;
        }
    }
    ...
}
```

lookup

JndiTemplate을 이용하여 argument 'name'으로 등록되어 있는 자원(resource)를 찾을 수 있다.

```
public Object lookupResource(final String name)
{
    try
    {
        return jndiTemplate.lookup(name);
    }
    catch (NamingException e)
    {
        e.printStackTrace();
        return null;
    }
}
```

lookup with requiredType

JndiTemplate의 lookup 메소드는 찾고자 하는 자원의 이름 뿐 아니라 원하는 타입(Type)을 지정할 수 있다.

```
public Foo lookupFoo(final String fooName)
{
    try
    {
        return jndiTemplate.lookup(fooName, Foo.class);
    }
    catch (NamingException e)
    {
        e.printStackTrace();
        return null;
    }
}
```

rebind

JndiTemplate의 rebind 메소드를 사용하여 자원을 재등록할 수 있다.




```
public boolean rebind(final String name, Object resource)
{
    try
    {
        jndiTemplate.rebind(name, resource);
        return true;
    }
    catch (NamingException e)
    {
        e.printStackTrace();
        return false;
    }
}
```

unbind

JndiTemplate의 unbind 메소드를 사용하여 등록된 자원을 등록해제할 수 있다.

```
public boolean unbind(final String name)
{
    try
    {
        jndiTemplate.unbind(name);
        return true;
    }
    catch (NamingException e)
    {
        e.printStackTrace();
        return false;
    }
}
```

참고자료

-  Spring Framework JndiTemplate class API
-  The Spring Framework - Reference Documentation A.2.3. The jee schema
-  Java SE Guide to JNDI

개요

Integration 서비스는 전자정부 표준프레임워크 기반의 시스템이 타 시스템과의 연계를 위해 사용하는 Interface의 표준을 정의한 것이다.

- Integration 서비스
 - 개요
 - 설명
 - 목적
 - 아키텍처
 - 구성
 - 참고자료

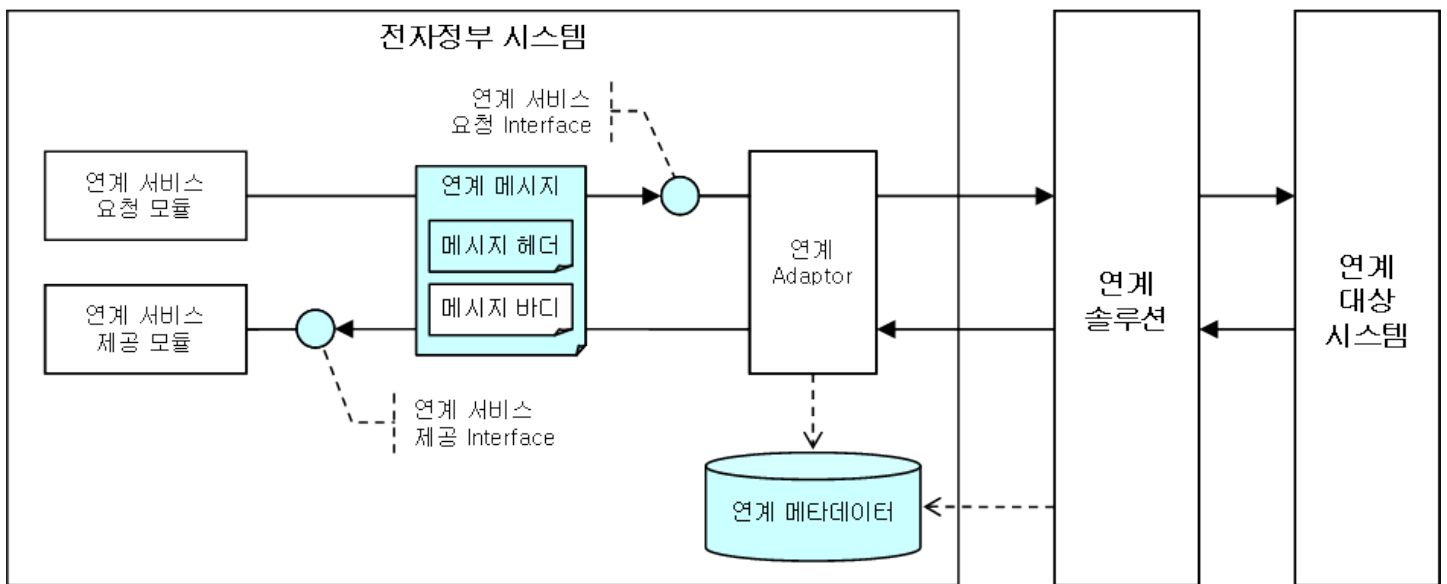
설명

목적

기존의 전자정부 시스템은 타 시스템과의 연계를 위해 연계 솔루션을 사용하거나 자체 개발한 연계 모듈을 사용해왔다. 기존에 사용된 연계 솔루션 및 자체 연계 모듈은 각각 고유한 설정 및 사용 방식을 가지고 있어, 동일한 연계 서비스라 할지라도 사용하는 연계 모듈에 따라 각기 다른 방식으로 개발되어 왔다. 본 Integration 서비스는 이러한 중복 개발을 없애고, 표준화된 설정 및 사용 방식을 정의하여 개발 효율성을 제고한다.

아키텍처

Integration 서비스를 사용하여 구현된 전자정부 시스템의 아키텍처는 다음과 같다.



Integration 서비스는 연계 서비스 요청 Interface, 연계 서비스 제공 Interface, 연계 메시지 및 메시지 헤더 등을 정의하고 있으며, 연계 서비스 요청 모듈 및 제공 모듈은 연계 Adaptor나 연계 솔루션과 관계 없이 Integration 서비스가 제공하는 Interface와 Class만을 사용하여 연계 업무를 수행할 수 있다. Integration 서비스는 연계 Interface 외에 연계에 필요한 정보를 담기 위한 Metadata를 정의하고 있다. Metadata는 연계 Interface를 사용하기 위해 필요한 최소한의 정보(연계 기관 정보, 연계 시스템 정보, 연계 서비스 정보, 메시지 형식 등)를 정의하고 있다.

구성

Integration 서비스는 연계에 필요한 정보를 정의한 Metadata와 연계 서비스를 사용 및 제공하기 위한 API로 구성된다.

- Metadata
- 연계 서비스 API

참고자료

N/A

Table of Contents

Metadata

개요

설명

논리모델

물리모델

Metadata Java Classes

ID 체계

참고자료

개요

Integration 서비스 Metadata는 연계에 필요한 정보를 정의하고 있다.
본 장은 실제 Integration 서비스로 구현된 연계 Adaptor를 사용하는 방식에 직접적인 도움을 주지는 않는다. 실제 사용법은 [연계 서비스 API](#)에서 설명하고 있다. 단, 연계 서비스 API의 핵심 Interface인 EgovIntegrationService의 단위에 해당하는 연계등록정보와, 이와 관련된 기관, 시스템, 서비스 등의 Metadata를 이해하는 것은 API 사용에 도움이 될 수 있다.

설명

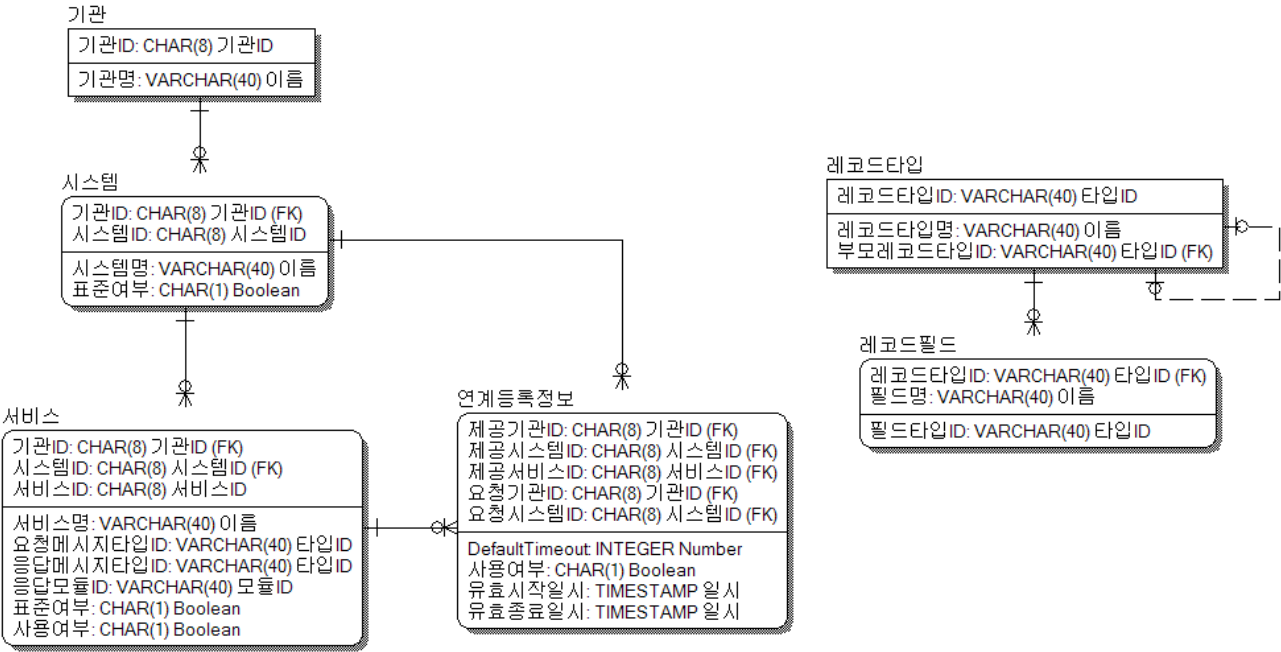
논리모델

Integration 서비스 Metadata의 논리모델은 연계를 위해 필요한 논리적인 정보를 정의한다.

논리ERD

Integration 서비스 Metadata의 논리ERD 및 Entity 설명은 다음과 같다.

- ERD의 Entity attribute의 Notation은 "<name> : <data type> <domain>" 이다.



Entity	설명
기관	연계 서비스를 제공 또는 사용하는 기관을 나타낸다. 하나의 기관은 다수의 시스템을 가지고 있다.
시스템	연계 서비스를 제공 또는 사용하는 시스템을 나타낸다. 하나의 시스템은 반드시 하나의 기관에 속하며, 다수의 서비스를 가지고 있다.
서비스	연계 서비스를 제공하는 단위를 나타낸다. 하나의 서비스는 반드시 하나의 시스템에 속한다.
연계등록정보	연계 서비스를 사용하기 위한 단위를 나타낸다. 연계 요청 시스템이 연계 제공 서비스를 사용하기 위해서 등록해야 하는 정보를 담고 있다.
레코드타입	연계에 사용되는 메시지의 형태를 나타낸다. <Key, Value> 쌍의 정보를 담고 있는 레코드 형태의 타입을 정의하고 있다. 하나의 레코드타입은 다수의 레코드필드를 가지고 있다.
레코드필드	레코드타입에 속하는 내부 필드의 정의를 나타낸다. 필드의 이름과 타입을 정의한다. 하나의 레코드필드는 반드시 하나의 레코드타입에 속한다.

논리모델 Domain 설명

Domain	Data Type	설명	비고
기관ID	CHAR(8)	연계 서비스를 제공 또는 사용하는 기관의 ID를 나타낸다.	ID 체계 참조
시스템ID	CHAR(8)	연계 서비스를 제공 또는 사용하는 시스템의 ID를 나타낸다.	ID 체계 참조
서비스ID	CHAR(8)	연계 서비스의 ID를 나타낸다.	ID 체계 참조
타입ID	VARCHAR(40)	메시지를 구성하는 각 구성 요소의 형식인 타입의 ID를 나타낸다.	ID 체계 참조
모듈ID	VARCHAR(40)	연계 서비스를 제공하기 위해 등록되어 있는 서비스 제공 모듈의 ID를 나타낸다.	ID 체계 참조
Boolean	CHAR(1)	참/거짓(true/false) 등의 논리값을 나타낸다.	'Y' : true 'N' : false
Number	INTEGER	일반적인 정수를 나타낸다.	
일시	DATETIME	일자와 시각을 포함하는 일시를 나타낸다.	
이름	VARCHAR(40)	기관, 시스템, 서비스 등의 각종 이름을 나타낸다.	

논리모델 Entity 설명

기관

Entity 명		기관			
설명		연계 서비스를 제공 또는 사용하는 기관을 나타낸다. 하나의 기관은 다수의 시스템을 가지고 있다.			
Attribute					
Seq	PK	Attribute명	Domain	Data Type	설명
1	Y	기관ID	기관ID	CHAR(8)	기관의 ID이다.
2		기관명	이름	VARCHAR(40)	기관의 이름이다.

시스템

Entity 명		시스템			
설명		연계 서비스를 제공 또는 사용하는 시스템을 나타낸다. 하나의 시스템은 반드시 하나의 기관에 속하며, 다수의 서비스를 가지고 있다.			
Attribute					
Seq	PK	Attribute명	Domain	Data Type	설명
1	Y	기관ID	기관ID	CHAR(8)	기관의 ID이다.
2	Y	시스템ID	시스템ID	CHAR(8)	시스템의 ID이다.
3		시스템명	이름	VARCHAR(40)	시스템의 이름이다.
4		표준여부	Boolean	CHAR(1)	표준 준수 여부를 나타낸다.

서비스

Entity 명		서비스			
설명		연계 서비스를 제공하는 단위를 나타낸다. 하나의 서비스는 반드시 하나의 시스템에 속한다.			
Attribute					
Seq	PK	Attribute명	Domain	Data Type	설명
1	Y	기관ID	기관ID	CHAR(8)	기관의 ID이다.
2	Y	시스템ID	시스템ID	CHAR(8)	시스템의 ID이다.
3	Y	서비스ID	서비스ID	CHAR(8)	서비스의 ID이다.
4		서비스명	이름	VARCHAR(40)	서비스의 이름이다.
5		요청메시지타입ID	타입ID	VARCHAR(40)	요청 메시지의 타입 ID이다.
6		응답메시지타입ID	타입ID	VARCHAR(40)	응답 메시지의 타입 ID이다.
7		응답모듈ID	모듈ID	VARCHAR(40)	실제 서비스를 제공하는 응답 모듈의 ID이다.
8		표준여부	Boolean	CHAR(1)	표준 준수 여부를 나타낸다.
9		사용여부	Boolean	CHAR(1)	서비스의 사용 여부를 나타낸다.

연계등록정보

Entity 명		연계등록정보			
설명		연계 서비스를 사용하기 위한 단위를 나타낸다. 연계 요청 시스템이 연계 제공 서비스를 사용하기 위해서 등록해야 하는 정보를 담고 있다.			
Attribute					
Seq	PK	Attribute명	Domain	Data Type	설명
1	Y	제공기관ID	기관ID	CHAR(8)	서비스를 제공하는 기관의 ID이다.
2	Y	제공시스템ID	시스템ID	CHAR(8)	서비스를 제공하는 시스템의 ID이다.
3	Y	제공서비스ID	서비스ID	CHAR(8)	서비스를 제공하는 서비스의 ID이다.
4	Y	요청기관ID	기관ID	CHAR(8)	서비스를 요청하는 기관의 ID이다.
5	Y	요청시스템ID	시스템ID	CHAR(8)	서비스를 요청하는 시스템의 ID이다.
6		DefaultTimeout	Number	INTEGER	서비스 요청 시 사용되는 default timeout 값이다.
7		사용여부	Boolean	CHAR(1)	등록된 연계의 사용 여부를 나타낸다.
8		유효시작일시	일시	DATEIME	등록된 연계가 유효한 기간의 시작 일시를 나타낸다.
9		유효종료일시	일시	DATEIME	등록된 연계가 유효한 기간의 종료 일시를 나타낸다.

레코드타입

Entity 명		레코드타입			
설명		연계에 사용되는 메시지의 형태를 나타낸다. <Key, Value> 쌍의 정보를 담고 있는 레코드 형태의 타입을 정의하고 있다. 하나의 레코드타입은 다수의 레코드필드를 가지고 있다.			
Attribute					
Seq	PK	Attribute명	Domain	Data Type	설명
1	Y	레코드타입ID	타입ID	VARCHAR(40)	레코드 타입의 ID이다.
2		레코드타입명	이름	VARCHAR(40)	레코드 타입의 이름이다.
3		부모레코드타입ID	타입ID	VARCHAR(40)	부모 레코드 타입의 ID이다.

레코드필드

Entity 명		레코드필드			
설명		레코드타입에 속하는 내부 필드의 정의를 나타낸다. 필드의 이름과 타입을 정의한다. 하나의 레코드필드는 반드시 하나의 레코드타입에 속한다.			
Attribute					
Seq	PK	Attribute명	Domain	Data Type	설명
1	Y	레코드타입ID	타입ID	VARCHAR(40)	필드가 속한 레코드의 타입 ID이다.
2	Y	필드명	이름	VARCHAR(40)	필드의 이름이다.
3		필드타입ID	타입ID	VARCHAR(40)	필드의 타입 ID이다.

Integration 서비스 Metadata의 물리모델은 논리모델을 실제 물리적인 DB로 구현하기 위한 모델로서, 아래 물리ERD는 Oracle DB를 가정하여 작성된 것이다. 물리모델은 Hibernate 등과 같은 Object Relational Mapping(ORM)을 사용하여 Access하는 것을 고려하여, 복수의 Attribute를 Identifier로 갖는 Entity를 Table로 변환할 때 Surrogate Key를 도입하고, 기존 Identifier는 Unique Constraints로 적용하여 정의되었다.

물리ERD

Integration 서비스 Metadata의 물리ERD 및 Table 설명은 다음과 같다.

- ERD의 Table Column의 Notation은 "<name> : <data type> <domain> <null option> <key>*" 이다.)
- 물리ERD의 경우, 각 연계 Adaptor 또는 연계 솔루션, 또는 시스템에 따라 사용하는 DB가 달라지므로 Data Type 등이 변경될 수 있다.

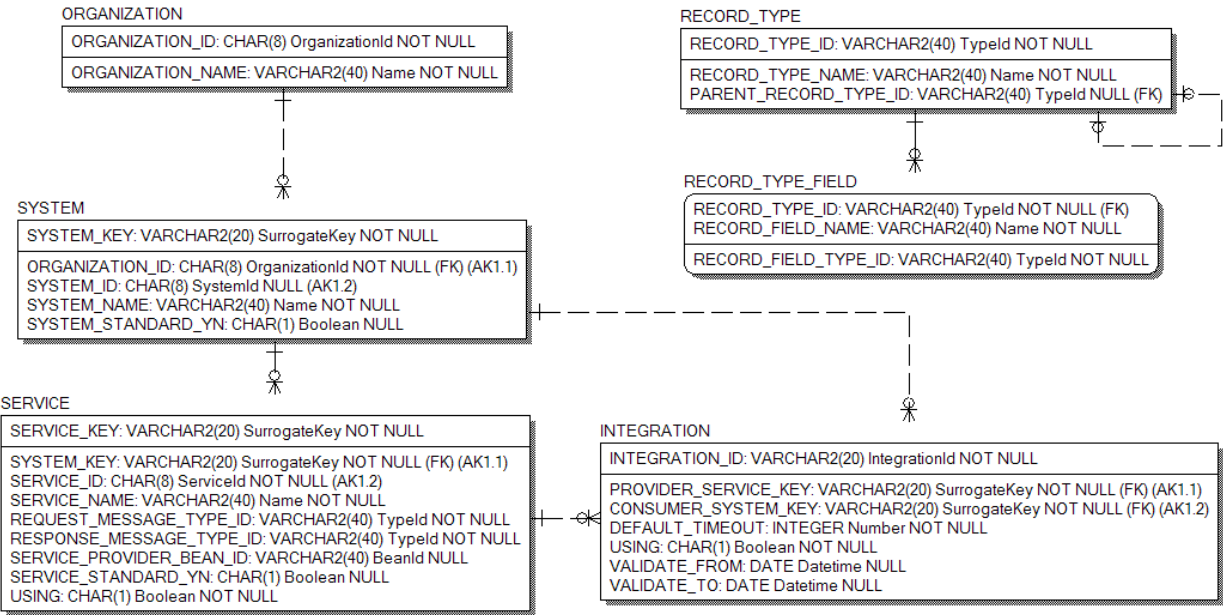


Table	Entity	설명
ORGANIZATION	기관	연계 서비스를 제공 또는 사용하는 기관을 나타낸다. 하나의 기관은 다수의 시스템을 가지고 있다.
SYSTEM	시스템	연계 서비스를 제공 또는 사용하는 시스템을 나타낸다. 하나의 시스템은 반드시 하나의 기관에 속하며, 다수의 서비스를 가지고 있다.
SERVICE	서비스	연계 서비스를 제공하는 단위를 나타낸다. 하나의 서비스는 반드시 하나의 시스템에 속한다.
INTEGRATION	연계등록정보	연계 서비스를 사용하기 위한 단위를 나타낸다. 연계 요청 시스템이 연계 제공 서비스를 사용하기 위해서 등록해야 하는 정보를 담고 있다.
RECORD_TYPE	레코드타입	연계에 사용되는 메시지의 형태를 나타낸다. <Key, Value> 쌍의 정보를 담고 있는 레코드 형태의 타입을 정의하고 있다. 하나의 레코드타입은 다수의 레코드필드를 가지고 있다.
RECORD_TYPE_FIELD	레코드필드	레코드타입에 속하는 내부 필드의 정의를 나타낸다. 필드의 이름과 타입을 정의한다. 하나의 레코드필드는 반드시 하나의 레코드타입에 속한다.

물리 모델 Domain 설명

Domain	Data Type	설명	비고
OrganizationId	CHAR(8)	연계 서비스를 제공 또는 사용하는 기관의 ID를 나타낸다.	ID 체계 참조
SystemId	CHAR(8)	연계 서비스를 제공 또는 사용하는 시스템의 ID를 나타낸다.	ID 체계 참조
ServiceId	CHAR(8)	연계 서비스의 ID를 나타낸다.	ID 체계 참조
Typeld	VARCHAR2(40)	메시지를 구성하는 각 구성 요소의 형식인 타입의 ID를 나타낸다.	ID 체계 참조
BeanId	VARCHAR2(40)	연계 서비스를 제공하기 위해 등록되어 있는 서비스 제공 모듈의 ID를 나타낸다.	ID 체계 참조
Boolean	CHAR(1)	참/거짓(true/false) 등의 논리값을 나타낸다.	'Y' : true 'N' : false
Number	INTEGER	일반적인 정수를 나타낸다.	
Datetime	DATEIME	일자와 시각을 포함하는 일시를 나타낸다.	
Name	VARCHAR2(40)	기관, 시스템, 서비스 등의 각종 이름을 나타낸다.	
SurrogateKey	VARCHAR2(20)	Composite 형태의 Primary Key를 대체하기 위한 키	

물리 모델 Table 설명

ORGANIZATION

Table 명		ORGANIZATION	Entity	기관			
설명		연계 서비스를 제공 또는 사용하는 기관을 나타낸다. 하나의 기관은 다수의 시스템을 가지고 있다.					
Column							
Seq	PK	Column명	한글명	Domain	Data Type	Null	설명
1	Y	ORGANIZATION_ID	기관ID	OrganizationId	CHAR(8)	N	기관의 ID이다.
2		ORGANIZATION_NAME	기관명	Name	VARCHAR2(40)	N	기관의 이름이다.

Constraints
PRIMARY KEY (ORGANIZATION_ID)

SYSTEM

Table 명		SYSTEM		Entity	시스템		
설명		연계 서비스를 제공 또는 사용하는 시스템을 나타낸다. 하나의 시스템은 반드시 하나의 기관에 속하며, 다수의 서비스를 가지고 있다.					
Column							
Seq	PK	Column명	한글명	Domain	Data Type	Null	설명
1	Y	SYSTEM_KEY	시스템KEY	SurrogateKey	VARCHAR2(20)	N	시스템의 Surrogate Key이다.
2		ORGANIZATION_ID	기관ID	OrganizationId	CHAR(8)	N	기관의 ID이다.
3		SYSTEM_ID	시스템ID	SystemId	CHAR(8)	N	시스템의 ID이다.
4		SYSTEM_NAME	시스템명	Name	VARCHAR2(40)	N	시스템의 이름이다.
5		STANDARD_YN	표준여부	Boolean	CHAR(1)	N	표준 준수 여부를 나타낸다.
Constraints							
PRIMAERY KEY (SYSTEM_KEY)							
UNIQUE (ORGANIZATION_ID, SYSTEM_ID)							
FOREIGN KEY (ORGANIZATION_ID) REFERENCES ORGANIZATION (ORGANIZATION_ID)							

SERVICE

Table 명		SERVICE	Entity	서비스			
설명		연계 서비스를 제공하는 단위를 나타낸다. 하나의 서비스는 반드시 하나의 시스템에 속한다.					
Column							
Seq	PK	Column명	한글명	Domain	Data Type	Null	설명
1	Y	SERVICE_KEY	서비스KEY	SurrogateKey	VARCHAR2(20)	N	서비스의 Surrogate Key이다.
2		SYSTEM_KEY	시스템KEY	SurrogateKey	VARCHAR2(20)	N	시스템의 Surrogate Key이다.
3		SERVICE_ID	서비스ID	ServiceId	CHAR(8)	N	서비스의 ID이다.
4		SERVICE_NAME	서비스명	Name	VARCHAR2(40)	N	서비스의 이름이다.
5		REQUEST_MESSAGE_TYPE_ID	요청메시지타입ID	TypeId	VARCHAR2(40)	N	요청 메시지의 타입 ID이다.
6		RESPONSE_MESSAGE_TYPE_ID	응답메시지타입ID	TypeId	VARCHAR2(40)	N	응답 메시지의 타입 ID이다.
7		SERVICE_PROVIDER_BEAN_ID	응답모듈ID	BeanId	VARCHAR2(40)	Y	실제 서비스를 제공하는 응답 모듈의 ID이다.
8		STANDARD_YN	표준여부	Boolean	CHAR(1)	N	표준 준수 여부를 나타낸다.
9		USING_YN	사용여부	Boolean	CHAR(1)	N	서비스의 사용 여부를 나타낸다.
Constraints							
PRIMARY KEY (SERVICE_KEY)							
UNIQUE (SYSTEM_KEY, SERVICE_ID)							
FOREIGN KEY (SYSTEM_KEY) REFERENCES SYSTEM (SYSTEM_KEY)							

INTEGRATION

Table 명		INTEGRATION	Entity	연계등록정보			
설명		연계 서비스를 사용하기 위한 단위를 나타낸다. 연계 요청 시스템이 연계 제공 서비스를 사용하기 위해서 등록해야 하는 정보를 담고 있다.					
Column							
Seq	PK	Column명	한글명	Domain	Data Type	Null	설명
1	Y	INTEGRATION_ID	연계ID	Surrogate Key	VARCHAR2(20)	N	연계등록정보의 Surrogate Key이다.
2		PROVIDER_SERVICE_KEY	제공서비스KEY	Surrogate Key	VARCHAR2(20)	N	서비스를 제공하는 서비스의 Surrogate Key이다.
3		CONSUMER_SYSTEM_KEY	요청시스템KEY	Surrogate Key	VARCHAR2(20)	N	서비스를 요청하는 시스템의 Surrogate Key이다.
4		DEFAULT_TIMEOUT	DefaultTimeout	Number	INTEGER	N	서비스 요청 시 사용되는 default timeout 값이다.
5		USING_YN	사용여부	Boolean	CHAR(1)	N	등록된 연계의 사용 여부를 나타낸다.
6		VALIDATE_FROM	유효시작일시	Datetime	DATETIME	Y	등록된 연계가 유효한 기간의 시작 일시를 나타낸다.
7		VALIDATE_TO	유효종료일시	Datetime	DATETIME	Y	등록된 연계가 유효한 기간의 종료 일시를 나타낸다.
Constraints							
PRIMARY KEY (INTEGRATION_ID)							
UNIQUE (PROVIDER_SERVICE_KEY, CONSUMER_SYSTEM_KEY)							
FOREIGN KEY (PROVIDER_SERVICE_KEY) REFERENCES SERVICE (SERVICE_KEY)							
ROREIGN KEY (CONSUMER_SYSTEM_KEY) REFERENCES SYSTEM (SYSTEM_KEY)							

RECORD_TYPE

Table 명		RECORD_TYPE	Entity	레코드타입			
설명		연계에 사용되는 메시지의 형태를 나타낸다. <Key, Value> 쌍의 정보를 담고 있는 레코드 형태의 타입을 정의하고 있다. 하나의 레코드타입은 다수의 레코드필드를 가지고 있다.					
Column							
Seq	PK	Column명	한글명	Domain	Data Type	Null	설명
1	Y	RECORD_TYPE_ID	레코드타입ID	TypeId	VARCHAR2(40)	N	레코드 타입의 ID이다.
2		RECORD_TYPE_NAME	레코드타입명	Name	VARCHAR2(40)	N	레코드 타입의 이름이다.
3		PARENT_RECORD_TYPE_ID	부모레코드타입ID	TypeId	VARCHAR2(40)	Y	부모 레코드 타입의 ID이다.
Constraints							
PRIMARY KEY (RECORD_TYPE_ID)							
FOREIGN KEY (PARENT_RECORD_TYPE_ID) REFERENCES RECORD_TYPE (RECORD_TYPE_ID)							

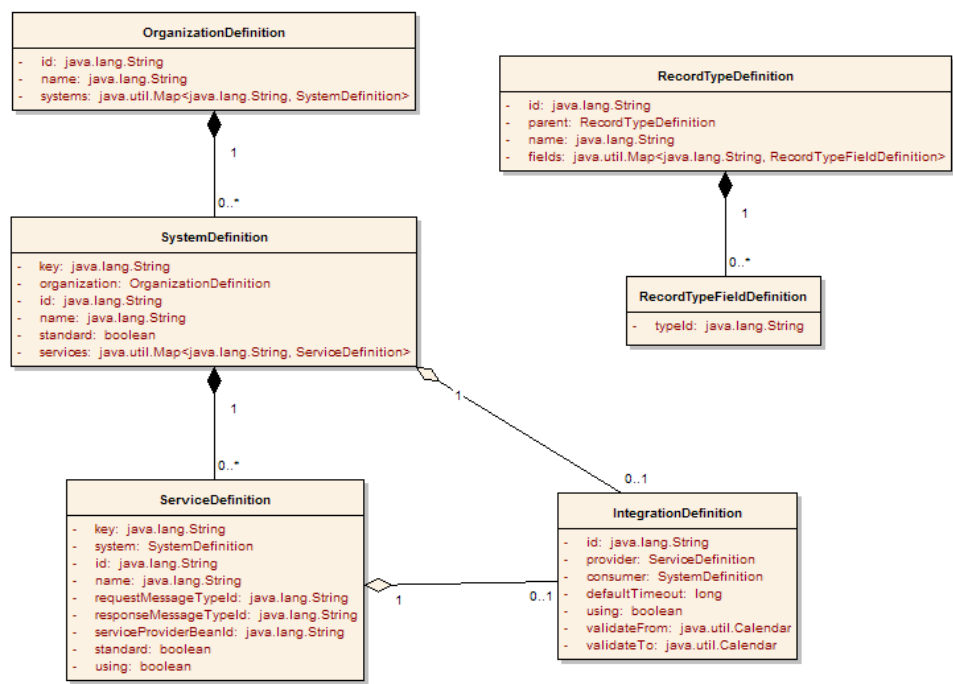
RECORD_TYPE_FIELD

Table 명		RECORD_TYPE_FIELD	Entity	레코드필드			
설명		레코드타입에 속하는 내부 필드의 정의를 나타낸다. 필드의 이름과 타입을 정의한다. 하나의 레코드필드는 반드시 하나의 레코드타입에 속한다.					
Column							
Seq	PK	Column명	한글명	Domain	Data Type	Null	설명
1	Y	RECORD_TYPE_ID	레코드타입ID	TypeId	VARCHAR2(40)	N	필드가 속한 레코드의 타입 ID이다.
2	Y	RECORD_FIELD_NAME	필드명	Name	VARCHAR2(40)	N	필드의 이름이다.
3		RECORD_FIELD_TYPE_ID	필드타입ID	TypeId	VARCHAR2(40)	N	필드의 타입 ID이다.
Constraints							
PRIMARY KEY (RECORD_TYPE_ID, RECORD_FIELD_NAME)							
FOREIGN KEY (RECORD_TYPE_ID) REFERENCES RECORD_TYPE (RECORD_TYPE_ID)							

Metadata Java Classes

Integration 서비스의 Metadata를 읽어오기 위한 Java Class를 정의하고 있다.

ClassDiagram



ID 체계

Integration 서비스 Metadata의 ID 체계는 다음과 같다.

ID	영문명	Data Type	형태	제약사항
기관 ID	OrganizationId	CHAR(8)	기관을 의미하는 Prefix 'ORG' 이후에 숫자 '0'으로 채워진 5자리 정수 Ex) ORG00000, ORG00001, ORG99999	모든 기관은 반드시 Unique한 ID를 가져야 한다.
시스템 ID	SystemId	CHAR(8)	시스템을 의미하는 Prefix 'SYS' 이후에 숫자 '0'으로 채워진 5자리 정수 Ex) SYS00000, SYS00001, SYS99999	동일한 기관에 속한 모든 시스템은 반드시 Unique한 ID를 가져야 한다. (서로 다른 기관에 속한 시스템은 같은 ID를 가질 수 있다.)
서비스 ID	ServiceId	CHAR(8)	서비스를 의미하는 Prefix 'SRV' 이후에 숫자 '0'으로 채워진 5자리 정수 Ex) SRV00000, SRV00001, SRV99999	동일한 시스템에 속한 모든 서비스는 반드시 Unique한 ID를 가져야 한다. (서로 다른 시스템에 속한 시스템은 같은 ID를 가질 수 있다.)
타입 ID	TypeId	VARCHAR2(40)	알파벳 대소문자, 숫자, '_' 로 구성된 길이 1 ~ 40 사이의 문자열 Ex) messageType0000, record_ABC	모든 타입은 Unique한 ID를 가져야 한다. 메타데이터로 저장되는 레코드타입의 ID는 다음의 Reserved 값을 가질 수 없다.boolean, string, byte, short, integer, long, biginteger, float, double, bigdecimal, calendar
모듈 ID	BeanId	VARCHAR2(40)	알파벳 대소문자, 숫자, '_' 로 구성된 길이 1 ~ 40 사이의 문자열 Ex) serviceProviderA, bean_123	모듈ID는 전자정부 개발프레임워크의 기반이 되는 Spring Framework에 등록되는 Bean Id이다.

참고자료

N/A

연계 서비스 API

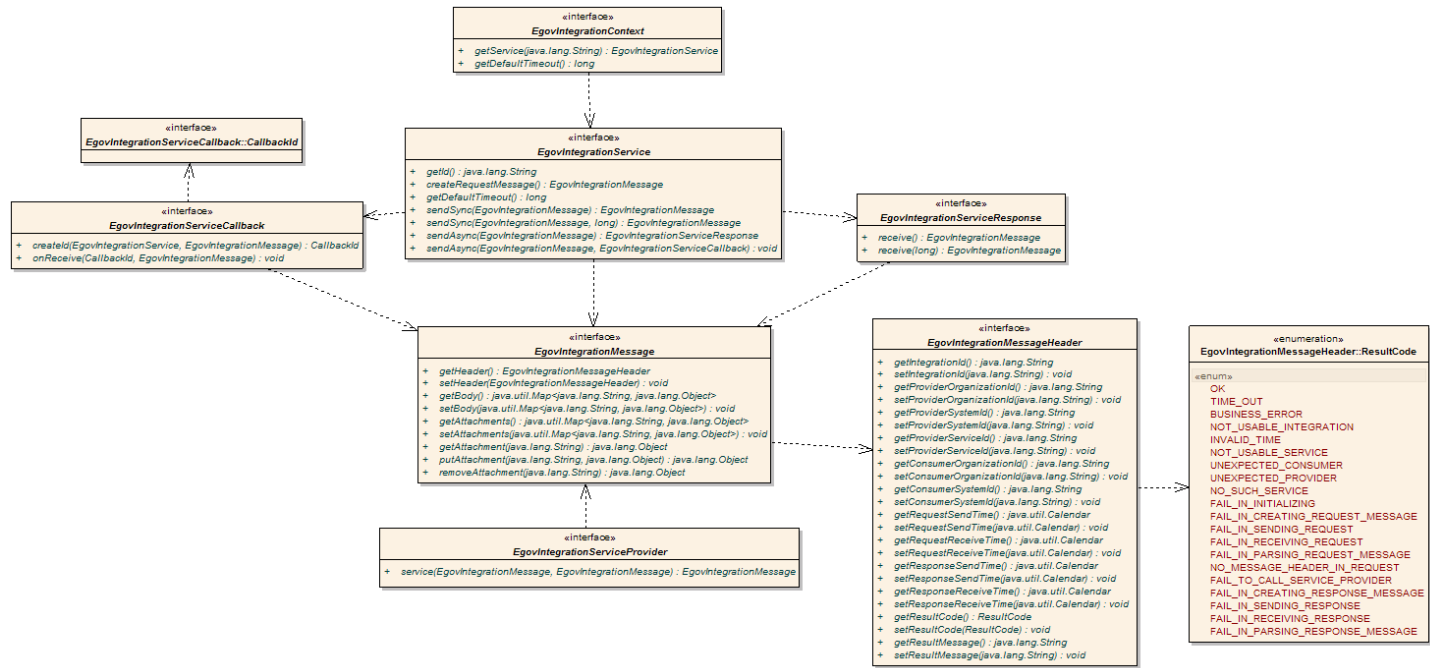
개요

연계 서비스 API는 연계 서비스를 사용 및 제공하기 위한 interface를 제공한다.

설명

구성

연계 서비스 API는 다음과 같이 구성된다.



구성요소	설명
EgovIntegrationContext	연계 서비스에 대한 설정 및 EgovIntegrationService 객체를 관리한다.
EgovIntegrationMessage	연계 서비스를 통해 주고 받는 표준 메시지를 정의한다.
EgovIntegrationMessageHeader	연계 서비스를 통해 주고 받는 표준 메시지 헤더를 정의한다.
EgovIntegrationMessageHeader::ResultCode	연계 서비스 결과 코드를 담고 있는 enumeration이다.
EgovIntegrationService	연계 서비스를 호출하기 위해 사용한다.
EgovIntegrationResponse	연계 서비스를 비동기 방식으로 호출한 경우, 응답 메시지를 받기 위해 사용한다.
EgovIntegrationServiceCallback	연계 서비스를 비동기 방식으로 호출한 경우, 응답 메시지를 받기 위한 Callback interface이다.
EgovIntegrationServiceCallback::CallbackId	연계 서비스를 Callback을 이용한 비동기 방식으로 호출한 경우, 요청 메시지와 응답 메시지를 연결하기 위한 ID를 나타내는 interface이다.
EgovIntegrationServiceProvider	연계 서비스를 제공하기 위해 사용한다.

EgovIntegrationContext

EgovIntegrationContext는 연계 서비스에 대한 설정 및 EgovIntegrationService 객체를 관리한다. 연계 서비스를 사용하기 위해서는 EgovIntegrationContext의 getService 메소드를 사용하여 EgovIntegrationService 객체를 얻어와야 한다. 아래는 주민등록번호와 성명을 이용하여 실명확인을 수행하는 예제이다.

```
package itl.sample;
import javax.annotation.Resource;
import egovframework.rte.itl.integration.EgovIntegrationContext;
import egovframework.rte.itl.integration.EgovIntegrationService;
public class EgovIntegrationSample {
    @Resource(name = "egovIntegrationContext")
    private EgovIntegrationContext egovIntegrationContext;
    public boolean verifyName(final String name, final String residentRegistrationNumber) {
        // 연번2가 "INT_VERIFY_NAME"인 연계 서비스 객체만 얻어온다.
        EgovIntegrationService service = egovIntegrationContext.getService("INT_VERIFY_NAME");
        // 요청 메시지 생성
        EgovIntegrationMessage requestMessage = service.createRequestMessage();
        // 요청 메시지 작성
        requestMessage.getBody().put("name", name);
        requestMessage.getBody().put("residentRegistrationNumber", residentRegistrationNumber);
        // 서비스 요청
        EgovIntegrationMessage responseMessage = service.sendSync(requestMessage);
        return responseMessage.getBody().get("result");
    }
}
```

위 예제에 해당하는 Metadata는 아래와 같다.

INTEGRATION						
ID	PROVIDER_SERVICE_KEY	CONSUMER_SYSTEM_KEY	DEFAULT_TIMEOUT	USING_YN	VALIDATE_FROM	VALIDATE_TO
'INT_VERIFY_NAME'	'SERVICE_VERIFY_NAME'	'SYSTEM_CONSUMER'	5000	'Y'	NULL	NULL

ORGANIZATION	
ID	NAME
'ORG00001'	'요청 기관'
'ORG00002'	'제공 기관'

SYSTEM				
SYSTEM_KEY	ORGANIZATION_ID	SYSTEM_ID	SYSTEM_NAME	STANDARD_YN
'SYSTEM_CONSUMER'	'ORG00001'	'SYS00001'	'요청 시스템'	'Y'
'SYSTEM_PROVIDER'	'ORG00002'	'SYS00001'	'응답 시스템'	'Y'

SERVICE								
SERVICE_KEY	SYSTEM_KEY	SERVICE_ID	SERVICE_NAME	REQUEST_MESSAGE_TYPE_ID	RESPONSE_MESSAGE_TYPE_ID	SERVICE_PROVIDER_BEAN_ID	USING_YN	STANDARD_YN
'SERVICE_VERIFY_NAME'	'SYSTEM_PROVIDER'	'SRV00001'	'VerifyName'	'REQ_VERIFY_NAME'	'RES_VERIFY_NAME'	'serviceVerifyName'	'Y'	'Y'

RECORD_TYPE		
RECORD_TYPE_ID	RECORD_TYPE_NAME	PARENT_RECORD_TYPE_ID
'REQ_VERIFY_NAME'	'RequestVerifyName'	NULL
'RES_VERIFY_NAME'	'ResponseVerifyName'	NULL

RECORD_TYPE_FIELD		
RECORD_TYPE_ID	RECORD_FIELD_NAME	RECORD_FIELD_TYPE_ID

Table of Contents

- 연계 서비스 API
 - 개요
 - 설명
 - 구성
 - EgovIntegrationContext
 - EgovIntegrationMessage
 - EgovIntegrationMessageHeader
 - EgovIntegrationMessageHeader::ResultCode
 - EgovIntegrationService
 - EgovIntegrationResponse
 - EgovIntegrationServiceProvider
 - 참고자료

'REQ_VERIFY_NAME'	'name'	'string'
'REQ_VERIFY_NAME'	'residentRegistrationNumber'	'string'
'RES_VERIFY_NAME'	'result'	'boolean'

EgovIntegrationMessage

EgovIntegrationMessage는 **헤더부**, **바디부**, **첨부파일**로 구성된다.

헤더부

EgovIntegrationMessage의 헤더부를 access 하기 위한 메소드는 아래와 같다.

Method Summary	
EgovIntegrationMessageHeader	getHeader()
void	setHeader(EgovIntegrationMessageHeader header)

바디부

EgovIntegrationMessage의 바디부를 access 하기 위한 메소드를 아래와 같다.

Method Summary	
Map<String, Object>	getBody()
void	setBody(Map<String, Object> body)

EgovIntegrationMessage의 바디부는 다음 값들로만 구성될 수 있다.

- Java Primitive Type의 Wrapper 객체(Boolean, Byte, Short, Integer, Long, Float, Double)
- BigInteger, BigDecimal
- String
- Calendar
- List<Object>
- Map<String, Object>

첨부파일

EgovIntegrationMessage의 첨부파일을 access 하기 위한 메소드를 아래와 같다.

Method Summary	
Map<String, Object>	getAttachments()
void	setAttachments(Map<String, Object> attachments)
Object	getAttachment(String name)
void	putAttachment(String name, Object attachment)
Object	removeAttachment(String name)

EgovIntegrationMessageHeader

EgovIntegrationMessageHeader는 다음과 같은 정보를 담고 있다.

Attribute Name	Data Type	설명
IntegrationId	String	연계ID
ProviderOrganizationId	String	연계 제공 기관ID
ProviderSystemId	String	연계 제공 시스템ID
ProviderServiceId	String	연계 제공 서비스ID
ConsumerOrganizationId	String	연계 요청 기관ID
ConsumerSystemId	String	연계 요청 시스템ID
RequestSendTime	Calendar	요청 송신 시각
RequestReceiveTime	Calendar	요청 수신 시각
ResponseSendTime	Calendar	응답 송신 시각
ResponseReceiveTime	Calendar	응답 수신 시각
ResultCode	ResultCode	결과 코드
ResultMessage	String	결과 메시지

EgovIntegrationMessageHeader는 위 attribute에 대한 get/set 메소드를 정의하고 있다.

EgovIntegrationMessageHeader::ResultCode

EgovIntegrationMessageHeader의 ResultCode Attribute는 다음 연계 서비스 결과 코드 중 하나의 값을 담고 있다.

Code Name	한글명	Value	설명
OK	정상 종료	"0000"	연계가 정상적으로 종료된 경우
TIME_OUT	Timeout 발생	"0001"	연계 수행 중 Client 단에서 Timeout이 발생한 경우
BUSINESS_ERROR	업무 오류 발생	"0002"	연계 수행 중 Server 단에서 업무적인 오류가 발생한 경우
NOT_USABLE_INTEGRATION	사용하지 않는 연계	"1000"	연계 정의(IntegrationDefinition)의 using flag가 false인 경우
INVALID_TIME	연계 가용시간이 아님	"1001"	연계를 요청한 시각이 연계 정의(IntegrationDefinition)의 validateFrom과 validateTo 사이가 아닌 경우 • 연계 가용시각 조건 = (validateFrom == null validateFrom.compareTo(now) <= 0) && (validateTo == null now.compareTo(validateTo) <= 0) * now : Calendaer = 현재 시각
NOT_USABLE_SERVICE	사용하지 않는 서비스	"1002"	연계 정의(IntegrationDefinition)에 등록된 제공 서비스(ServiceDefinition)의 using flag 값이 false인 경우
UNEXPECTED_CONSUMER	기대하지 않은 연계 요청자	"1003"	연계 제공 시스템(Server)에 등록된 연계 정의(IntegrationDefinition)의 요청 시스템 코드값과 요청 메시지 헤더의 요청 시스템 코드값이 일치하지 않는 경우
UNEXPECTED_PROVIDER	기대하지 않은 연계 제공자	"1004"	연계 제공 시스템(Server)에 등록된 시스템 코드와 요청 메시지 헤더의 제공 시스템 코드값이 일치하지 않는 경우
NO_SUCH_SERVICE	제공하지 않는 서비스	"1005"	연계 제공 시스템(Server)에 등록되어 있지 않은 서비스를 요청한 경우
FAIL_IN_INITIALIZING	연계 서비스 초기화 실패	"1006"	연계 제공 서비스를 초기화하는데 실패한 경우
FAIL_IN_CREATING_REQUEST_MESSAGE	요청 메시지 생성 실패	"2000"	Client에서 요청 메시지를 생성할 때 오류가 발생한 경우
FAIL_IN_SENDING_REQUEST	요청 메시지 송신 실패	"2001"	Client에서 요청 메시지를 송신할 때 오류가 발생한 경우
FAIL_IN_RECEIVING_REQUEST	요청 메시지 수신 실패	"3000"	Server에서 요청 메시지를 수신할 때 오류가 발생한 경우
FAIL_IN_PARSING_REQUEST_MESSAGE	요청 메시지 분석 실패	"3001"	Server에서 요청 메시지를 분석할 때 오류가 발생한 경우
NO_MESSAGE_HEADER_IN_REQUEST	요청 메시지 헤더 부재	"3002"	Server에서 받은 요청 메시지에 표준 메시지 헤더가 존재하지 않는 경우
FAIL_TO_CALL_SERVICE_PROVIDER	서비스 제공 모듈 호출 실패	"3003"	Server에서 서비스 제공 모듈을 호출할 때 오류가 발생한 경우
FAIL_IN_CREATING_RESPONSE_MESSAGE	응답 메시지 생성 실패	"4000"	Server에서 응답 메시지를 생성할 때 오류가 발생한 경우
FAIL_IN_SENDING_RESPONSE	응답 메시지 송신 실패	"4001"	Server에서 응답 메시지를 송신할 때 오류가 발생한 경우
FAIL_IN_RECEIVING_RESPONSE	응답 메시지 수신 실패	"5000"	Client에서 응답 메시지를 수신할 때 오류가 발생한 경우
FAIL_IN_PARSING_RESPONSE_MESSAGE	응답 메시지 분석 실패	"5001"	Client에서 응답 메시지를 분석할 때 오류가 발생한 경우

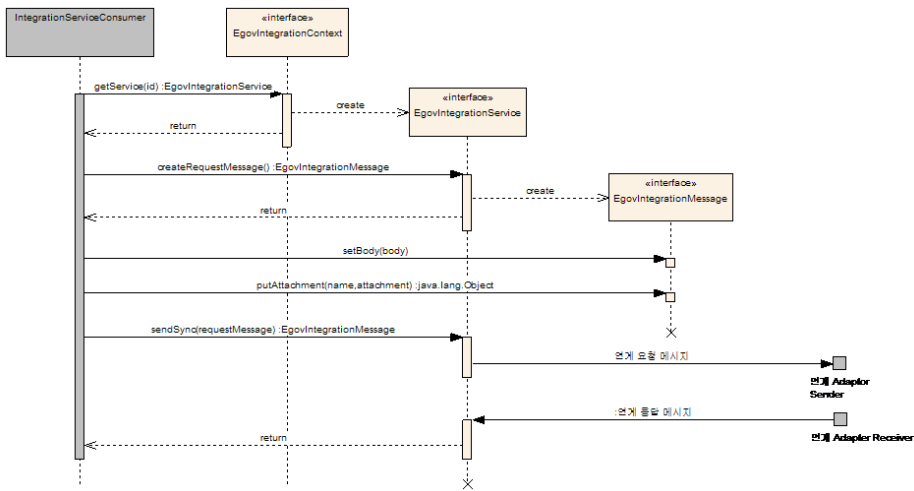
EgovIntegrationService

EgovIntegrationService를 동기화 방식의 호출과 비동기화 방식의 호출을 지원한다.

- 동기화 방식
- 비동기화 방식

동기화 방식

EgovIntegrationService의 sendSync 메소드는 동기화 방식으로 연계 서비스를 호출한다.



```

...
public class EgovIntegrationSample
{
    ...
    public boolean verifyName (final String name, final String residentRegistrationNumber)
    {
        // EgovIntegrationContext 에서 EgovIntegrationService 객체를 얻어온 후, 요청 메시지를 생성 및 작성한다.
        ...
        // 동기방식으로 연계 서비스 호출 (timeout = 5000 millisecond)
        EgovIntegrationMessage responseMessage = service.sendSync(requestMessage, 5000);
        // 응답 결과 처리
        ...
    }
    ...
}

```

EgovIntegrationContext 또는 Metadata의 연계등록정보에 등록된 default timeout 값을 사용할 경우, timeout 값을 생략할 수 있다.

```

...
EgovIntegrationMessage responseMessage = serviced.sendSync(requestMessage);
...

```

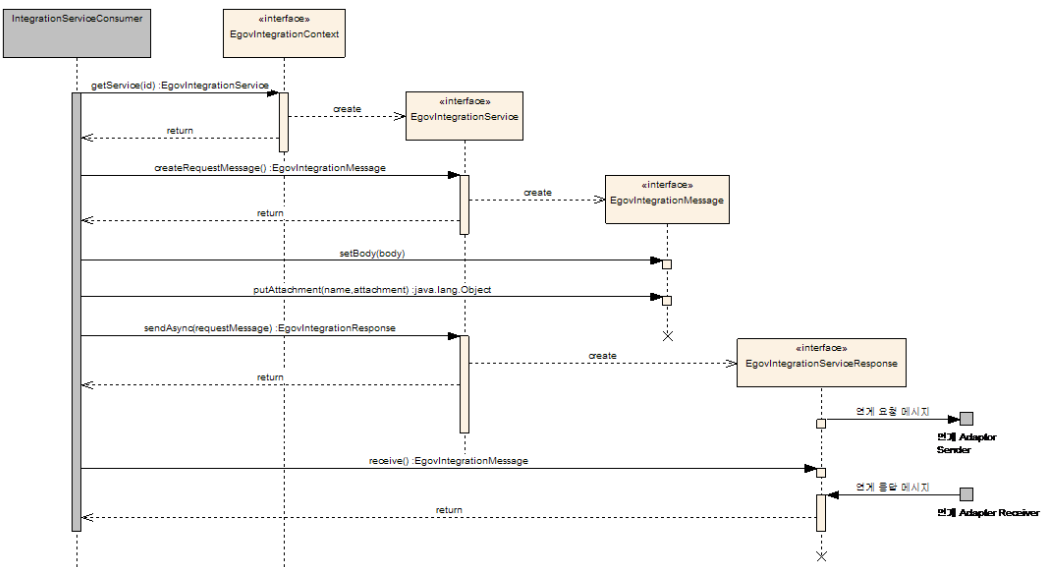
비동기화 방식

EgovIntegrationService의 sendAsync 메소드는 비동기화 방식으로 연계 서비스를 호출한다. sendAsync 메소드는 두가지 방식이 존재한다.

- Using sendAsync with Response
- Using sendAsync with Callback

Using sendAsync with Response

EgovIntegrationServiceResponse를 이용한 비동기 호출 방식이다. Response 방식의 비동기 호출은 연계 서비스를 요청하는 업무 모듈에 응답에 대한 ownership을 가지고 있으며, 응답 결과를 스스로 처리해야 하는 경우 사용한다.



```

...
public class EgovIntegrationSample
{
    ...
    public boolean verifyName (final String name, final String residentRegistrationNumber)
    {
        // EgovIntegrationContext 에서 EgovIntegrationService 객체를 얻어온 후, 요청 메시지를 생성 및 작성한다.
        ...
        // 비동기방식으로 연계 서비스 호출
        EgovIntegrationServiceResponse response = service.sendAsync(requestMessage);
        // response 객체를 이용하여 응답 메시지를 받기 전에 필요한 업무를 수행
        ...
        // response 객체를 이용하여 응답 메시지 수신(timeout = 5000 millisecond)
        EgovIntegrationMessage responseMessage = response.receive(5000);
        // 응답 메시지 처리
        ...
    }
    ...
}

```

EgovIntegrationContext 또는 Metadata의 연계등록정보에 등록된 default timeout 값을 사용할 경우, timeout 값을 생략할 수 있다.

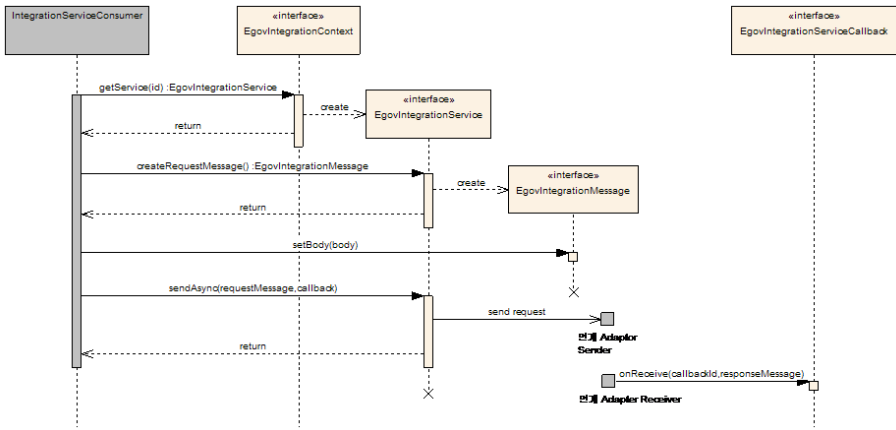
```

...
// response 객체를 이용하여 응답 메시지 수신
EgovIntegrationMessage responseMessage = response.receive();
...

```

Using sendAsync with Callback

EgovIntegrationServiceCallback를 이용한 비동기 호출 방식이다. Callback 방식의 비동기 호출은 연계 서비스를 요청하는 업무 모듈은 단지 요청만을 수행하고, 응답에 대한 처리는 Callback 객체에 위임해도 상관없는 경우 사용한다.



```
...
public class EgovIntegrationSample
{
    ...
    @Resource(name = "verifyNameServiceCallback")
    private EgovIntegrationServiceCallback callback;

    public boolean verifyName(final String name, final String residentRegistrationNumber)
    {
        // EgovIntegrationContext에서 EgovIntegrationService 객체를 얻은 후, 요청 메시지를 생성 및 작성한다.
        ...

        // 비동기방식으로 연계 서비스 호출
        service.sendSync(requestMessage, callback);
    }
    ...
}

package itl.sample;
import egovframework.rte.itl.integration.EgovIntegrationServiceCallback;
import egovframework.rte.itl.integration.EgovIntegrationServiceCallback.CallbackId;
public class VefiryNameServiceCallback
{
    public CallbackId createId(EgovIntegrationService service, EgovIntegrationMessage requestMessage)
    {
        // 본 메소드는 EgovIntegrationService를 구현한 연계 Adaptor 또는 송무선에서 불리워진다.
        // 서비스와 요청 메시지를 이용하여 CallbackId를 생성하여 return한다.
        // 생성된 CallbackId는 응답 메시지 수신 시, 해당하는 서비스 및 요청 메시지를 식별하기 위해 사용한다.
        // CallbackId 생성
        CallbackId callbackId = ...
        return callbackId;
    }

    public void onReceive(CallbackId callbackId, EgovIntegrationMessage responseMessage)
    {
        // 본 메소드는 처리해야 하는 응답 메시지가 도착했을 때, 연계 Adaptor 또는 송무선에 의해 불리워진다.
        // 응답 메시지 처리
        ...
    }
}
```

EgovIntegrationServiceProvider

EgovIntegrationServiceProvider interface는 연계 서비스를 제공하기 위한 interface로 연계 서비스를 제공하는 모듈은 본 interface를 implements 해야한다.
아래 예제는 이름과 주민등록번호는 이용하여 실명확인을 수행하는 서비스를 제공하는 업무 모듈과 Spring Framework Configuration XML 파일이다. (Metadata는 EgovIntegrationContext 예제의 설정과 같다.)

```
package itl.sample;
import egovframework.rte.itl.integration.EgovIntegrationMessage;
import egovframework.rte.itl.integration.EgovIntegrationServiceProvider;
public class ServiceVerifyName implements EgovIntegrationServiceProvider
{
    public void service(EgovIntegrationMessage requestMessage, EgovIntegrationMessage responseMessage)
    {
        String name = requestMessage.getBody().get("name");
        String residentRegistrationNumber = requestMessage.getBody().get("residentRegistrationNumber");
        // 실명 확인
        boolean result = verifyName(name, residentRegistrationNumber);
        responseMessage.getBody().put("result", result);
    }
}
```

```
...
<bean id="serviceVerifyName" class="itl.sample.ServiceVerifyName"/>
...
```

참고자료

N/A

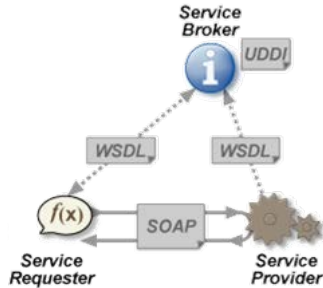
개요

WebService는 전자정부 개발프레임워크 Integration 서비스 표준에 따라 WebService를 요청하고 제공하기 위한 Library이다.

주요 개념

Web Services

W3C는 Web Service를 "네트워크 상에서 발생하는 컴퓨터 간의 상호작용을 지원하기 위한 소프트웨어 시스템"으로 정의하고 있다. 일반적으로 Web Service는 인터넷과 같은 네트워크 상에서 접근되고, 요청된 서비스를 제공하는 원격 시스템에서 수행되는 Web APIs이다.



참조 : http://en.wikipedia.org/wiki/Web_Services

사용 오픈소스

Apache CXF

WebService는 Web Service 구현하기 위해서 [Apache CXF](#)를 사용한다.

설명

WebService는 [Integration Service](#) 표준에 따라 구현한 Library이므로, 본 장에서는 API 등의 사용 방식은 설명하지 않는다. 본 장은 WebService 만을 위한 추가적인 설정 정보를 설명하고, 설정 방법을 가이드한다.

Metadata

WebService는 연계 서비스를 요청하고 제공하기 위한 Web Service Client와 Server 정보를 필요로 한다.

물리ERD

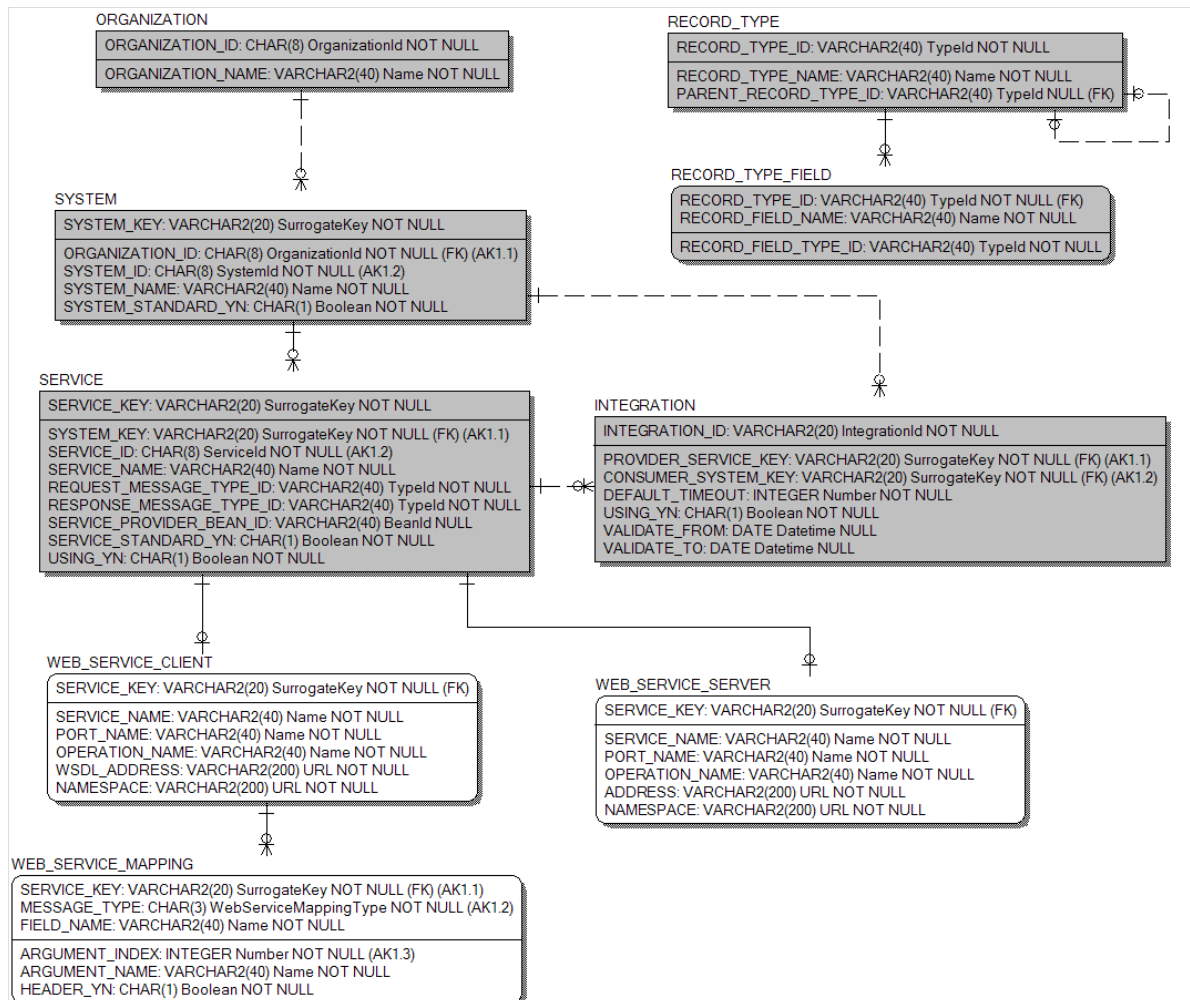


Table	설명
WEB_SERVICE_SERVER	연계 서비스를 Web Service 형태로 공개(publish)하기 위해 필요한 정보를 담고 있다.
WEB_SERVICE_CLIENT	Web Service 형태로 공개(publish)되어 있는 연계 서비스를 호출하기 위해 필요한 정보를 담고 있다.
WEB_SERVICE_MAPPING	전자정부 Integration 서비스 표준에 따라 개발된 서비스가 아닌 기존의 Legacy 시스템의 Web Service를 호출하기 위해, 표준 메시지와 Web Service 메시지 간의 mapping 정보를 담고 있다.

물리 모델 Domain 설명

Domain	Data Type	설명	비고
URL	VARCHAR2(200)	URL을 나타낸다.	
WebServiceMappingType	CHAR(2)	Req/Res 구분을 나타낸다.	'REQ' : Request 'RES' : Response

물리 모델 Table 설명

WEB_SERVICE_SERVER

Table 명		WEB_SERVICE_SERVER					
설명		연계 서비스를 Web Service 형태로 공개(publish)하기 위해 필요한 정보를 담고 있다.					
Column							
Seq	PK	Column명	한글명	Domain	Data Type	Null	설명
1	Y	SERVICE_KEY	서비스KEY	SurrogateKey	VARCHAR2(20)	N	서비스 Key이다.
2		ADDRESS	주소	URL	VARCHAR2(200)	N	서비스를 공개할 주소이다.
3		NAMESPACE	네임스페이스	URL	VARCHAR2(200)	N	서비스의 네임스페이스이다.
4		SERVICE_NAME	서비스명	Name	VARCHAR2(40)	N	공개할 때 사용할 서비스의 이름이다.
5		PORT_NAME	포트명	Name	VARCHAR2(40)	N	공개할 때 사용할 포트의 이름이다.
6		OPERATION_NAME	기능명	Name	VARCHAR2(40)	N	공개할 때 사용할 기능의 이름이다.
Constraints							
PRIMARY KEY (SERVICE_KEY)							
FOREIGN KEY (SERVICE_KEY) REFERENCES SERVICE (SERVICE_KEY)							

WEB_SERVICE_CLIENT

Table 명		WEB_SERVICE_CLIENT					
설명		Web Service 형태로 공개(publish)되어 있는 연계 서비스를 호출하기 위해 필요한 정보를 담고 있다.					
Column							
Seq	PK	Column명	한글명	Domain	Data Type	Null	설명
1	Y	SERVICE_KEY	서비스KEY	SurrogateKey	VARCHAR2(20)	N	서비스 Key이다.
2		WSDL_ADDRESS	WSDL 주소	URL	VARCHAR2(200)	N	사용할 서비스의 WSDL 주소이다.
3		NAMESPACE	네임스페이스	URL	VARCHAR2(200)	N	서비스의 네임스페이스이다.
4		SERVICE_NAME	서비스명	Name	VARCHAR2(40)	N	사용할 서비스의 이름이다.
5		PORT_NAME	포트명	Name	VARCHAR2(40)	N	사용할 포트의 이름이다.
6		OPERATION_NAME	기능명	Name	VARCHAR2(40)	N	사용할 기능의 이름이다.
Constraints							
PRIMARY KEY (SERVICE_KEY)							
FOREIGN KEY (SERVICE_KEY) REFERENCES SERVICE (SERVICE_KEY)							

WEB_SERVICE_MAPPING

Table 명		WEB_SERVICE_MAPPING					
설명		전자정부 Integration 서비스 표준에 따라 개발된 서비스가 아닌 기존의 Legacy 시스템의 Web Service를 호출하기 위해, 표준 메시지와 Web Service 메시지 간의 mapping 정보를 담고 있다.					
Column							
Seq	PK	Column명	한글명	Domain	Data Type	Null	설명
1	Y	SERVICE_KEY	서비스KEY	SurrogateKey	VARCHAR2(20)	N	서비스 Key이다.
2	Y	MESSAGE_TYPE	메시지타입	WebServiceMappingType	CHAR(3)	N	Req/Res 구분이다.
3	Y	FIELD_NAME	필드명	Name	VARCHAR2(40)	N	표준 메시지 Field 이름이다.
4		ARGUMENT_INDEX	변수순서	Number	Integer	N	Web Service 메시지의 변수 순서이다.
5		ARGUMENT_NAME	변수명	Name	VARCHAR2(40)	N	Web Service 메시지의 변수 이름이다.
6		HEADER_YN	헤더여부	Boolean	CHAR(1)	N	Web Service 헤더 여부이다.
Constraints							
PRIMARY KEY (SERVICE_KEY, MESSAGE_TYPE, FIELD_NAME)							
FOREIGN KEY (SERVICE_KEY) REFERENCES WEB_SERVICE_CLIENT (SERVICE_KEY)							

설정 방법

WebService를 사용하기 위해 다음의 설정이 필요하다.

1. pom.xml에 dependency 설정 추가
2. Spring XML Configuration 설정

pom.xml에 dependency 설정 추가

WebService를 사용하기 위해서 pom.xml의 dependencies tag에 다음 dependency를 추가한다.

설정 방법

WebService를 사용하기 위해 다음의 설정이 필요하다.

1. pom.xml에 dependency 설정 추가
2. Spring XML Configuration 설정

pom.xml에 dependency 설정 추가

WebService를 사용하기 위해서 pom.xml의 dependencies tag에 다음 dependency를 추가한다.

- <version> tag의 값인 \${egovframework.version}에는 사용할 egovframework의 version을 기재한다.

```
<dependencies>
...
<dependency>
  <groupId>egovframework.rte</groupId>
  <artifactId>egovframework.rte.itl.webservice</artifactId>
  <version>${egovframework.version}</version>
</dependency>
</dependencies>
...
```

Spring XML Configuration 설정

WebService를 위한 기본적인 설정이 포함된 "context-webservice.xml" 파일을 Spring XML Configuration 파일에 import한다.

```
<import resource="classpath:/egovframework/rte/itl/webservice/context/context-webservice.xml" />
```

그리고 Context와 DataSource를 등록해야 한다.(DataSource의 경우, 프로젝트에서 사용하는 것이 있을 경우 설정하지 않아도 된다. 단, 반드시 id가 "dataSource" 이여야 한다.)

```
<!-- EgovWebServiceContext 이다.
organizationId 와 systemId 는 현재 시스템의 기관ID 및 시스템ID를 넣어야 한다. -->
<bean id="egovWebServiceContext"
      class="egovframework.rte.itl.webservice.EgovWebServiceContext"
      init-method="init">
  <property name="organizationId" value="ORG_EGOV" />
  <property name="systemId" value="SYS00001" />
  <property name="defaultTimeout" value="5000" />
  <property name="integrationDefinitionDao" ref="integrationDefinitionDao" />
  <property name="webServiceServerDefinitionDao" ref="webServiceServerDefinitionDao" />
  <property name="webServiceClientDefinitionDao" ref="webServiceClientDefinitionDao" />
  <property name="typeLoader" ref="typeLoader" />
  <property name="classLoader" ref="classLoader" />
</bean>

<!-- DataSource 설정이다. 시스템에 맞게 제작성 해야 한다. 아래는 HSQL Sample이다. -->
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
  <property name="driverClassName" value="net.sf.log4jdbc.DriverSpy" />
  <property name="url" value="jdbc:log4jdbc:hsqldb:hsqldb://localhost/test" />
  <property name="username" value="sa" />
  <property name="password" value="" />
  <property name="defaultAutoCommit" value="false" />
  <property name="poolPreparedStatements" value="true" />
</bean>
```

Client 모듈 개발

WebService Client 모듈은 Web Service로 공개된 Integration 서비스 표준에 따라 호출하는 모듈로서, 본 장은 설정 방식을 설명한다. (호출 방식은 [연계 서비스 API](#)를 참조한다.) Client 모듈을 설정하기 위해서는 다음 과정이 필요하다.

1. Metadata WEB_SERVICE_CLIENT 설정 추가
2. (Optional) Metadata WEB_SERVICE_MAPPING 설정 추가

Metadata WEB_SERVICE_CLIENT 설정 추가

Client 모듈을 설정하기 위해서는 Metadata의 WEB_SERVICE_CLIENT Table에 설정을 추가해야 한다.

다음과 같이 Integration 서비스의 Metadata인 INTEGRATION Table에 연계등록정보가 설정되어 있다고 가정한다. (* 기관, 시스템, 서비스, 메시지타입 등의 정보는 설정되어 있으며, 개발하는 시스템은 'SYSTEM_CONSUMER'라고 가정함)

INTEGRATION						
ID	PROVIDER_SERVICE_KEY	CONSUMER_SYSTEM_KEY	DEFAULT_TIMEOUT	USING_YN	VALIDATE_FROM	VALIDATE_TO
'INT_VERIFY_NAME'	'SERVICE_VERIFY_NAME'	'SYSTEM_CONSUMER'	5000	'Y'	NULL	NULL

Web Service 'SERVICE_VERIFY_NAME'를 호출하기 위해서 WEB_SERVICE_CLIENT에 'SERVICE_VERIFY_NAME'을 SERVICE_KEY로 갖는 설정을 추가해야 한다.

WEB_SERVICE_CLIENT					
SERVICE_KEY	WSDL_ADDRESS	NAMESPACE	SERVICE_NAME	PORT_NAME	OPERATION_NAME
'SERVICE_VERIFY_NAME'	'http://192.168.0.1:8080/Sample/services/VerifyName?wsdl'	'http://it/sample/'	'VerifyNameService'	'VerifyNamePort'	'service'

(Optional) Metadata WEB_SERVICE_MAPPING 설정 추가

만약 호출하는 Web Service가 전자정부 Integration 서비스 표준에 따라 개발된 서비스가 아닌 경우, 메시지 헤더부가 다를 수 있어 별도의 Mapping 정보가 필요하다. 전자정부 Integration 서비스 표준은 Web Service Header부에 들어갈 Attribute들이 EgovIntegrationMessageHeader에 정의되어 있고, 바디부는 EgovIntegrationMessage의 body에 정의되어 있으므로 별도의 mapping 정보 없이 header와 body 부의 구분이 가능하지만, 표준을 따르지 않은 Web Service의 경우 EgovIntegrationMessage의 body부에 정의되어 있는 일부 값들을 헤더에 포함시켜야 한다.

WEB_SERVICE_MAPPING Table의 정보는 Integration 서비스 표준에 정의되어 있는 메시지 형태를 기준으로 한다. 서비스 'SERVICE_VERIFY_NAME'의 Request Message는 'name', 'residentRegistrationNumber' 필드를 가지고, Response Message는 'result' 필드를 가진다. 따라서 'SERVICE_VERIFY_NAME'에 해당하는 WEB_SERVICE_MAPPING은 다음의 정보를 가져야 한다.

WEB_SERVICE_MAPPING					
SERVICE_KEY	MESSAGE_TYPE	FIELD_NAME	ARGUMENT_INDEX	ARGUMENT_NAME	HEADER_YN
'SERVICE_VERIFY_NAME'	'REQ'	'name'	1	'name'	Y
'SERVICE_VERIFY_NAME'	'REQ'	'residentRegistrationNumber'	2	'residentRegistrationNumber'	N
'SERVICE_VERIFY_NAME'	'RES'	'result'	1	'result'	N

위 정보 중 HEADER_YN column의 값에 따라 해당 field가 Web Service Envelop의 header에 포함될지 여부를 판단한다. 위 설정값을 적용하면, 요청 메시지 중 'name' field는 Web Service Envelop의 헤더에 포함된다.

Server 모듈 개발

Web Service Server 모듈을 개발하는 과정은 다음과 같다.

- 1. web.xml에 EgovWebServiceServlet 추가
- 2. Metadata WEB_SERVICE_SERVER 설정 추가

web.xml에 EgovWebServiceServlet 추가

web.xml에 EgovWebServiceServlet 설정을 추가한다.

```
...
<servlet>
  <description></description>
  <display-name>EgovWebServiceServlet</display-name>
  <servlet-name>EgovWebServiceServlet</servlet-name>
  <servlet-class>egovframework.rte.itl.webservice.EgovWebServiceServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>EgovWebServiceServlet</servlet-name>
  <url-pattern>/services/*</url-pattern>
</servlet-mapping>
...
```

<url-pattern> tag의 값은 변경 될 수 있다. 자세한 설명은 다음 WEB_SERVICE_SERVER 설정을 참조한다.

Metadata WEB_SERVICE_SERVER 설정 추가

다음과 같이 Integration 서비스의 Metadata인 INTEGRATION Table에 연계등록정보가 설정되어 있다고 가정한다. (* 기관, 시스템, 서비스, 메시지타입 등의 정보는 설정되어 있으며, 공개할 서비스는 'SERVICE_VERIFY_NAME'이라고 가정함)

INTEGRATION						
ID	PROVIDER_SERVICE_KEY	CONSUMER_SYSTEM_KEY	DEFAULT_TIMEOUT	USING_YN	VALIDATE_FROM	VALIDATE_TO
'INT_VERIFY_NAME'	'SERVICE_VERIFY_NAME'	'SYSTEM_CONSUMER'	5000	'Y'	NULL	NULL

Web Service 'SERVICE_VERIFY_NAME'를 공개하기 위해서 WEB_SERVICE_SERVER에 'SERVICE_VERIFY_NAME'을 SERVICE_KEY로 갖는 설정을 추가해야 한다.

WEB_SERVICE_SERVICE					
SERVICE_KEY	ADDRESS	NAMESPACE	SERVICE_NAME	PORT_NAME	OPERATION_NAME
'SERVICE_VERIFY_NAME'	'/VerifyName'	'http://it/sample/'	'VerifyNameService'	'VerifyNamePort'	'service'

<servlet-mapping> tag의 <url-pattern> tag의 값은 서비스를 제공하기 위한 주소로, WEB_SERVICE_SERVER Table의 ADDRESS Column 값은 <url-pattern> tag값에 대한 상대 위치를 나타낸다. 예를 들어, Web Application의 IP가 192.168.0.1, Port가 8080, Context Root가 "Sample", url-patterns이 "/services/*"인 경우, 위 'SERVICE_VERIFY_NAME'의 WSDL Address는 http://192.168.0.1:8080/Sample/services/VerifyName?wsdl이다.

WAS에 배포

전자정부 Webservice를 포함한 어플리케이션을 WAS에 배포(deploy)하는 방법을 설명한다. Apache CXF의 경우 Web Service 관련 라이브러리를 CXF에서 제공하는 것을 사용해야 한다. 만약 WAS가 기본적으로 Web Service 라이브러리를 제공할 경우, 정상적으로 동작하지 않을 수 있다. 따라서 CXF 라이브러리를 사용할 수 있도록 설정을 변경해야 하는데, 대부분의 해결책은 Web Application의 WEB-INF의 라이브러리를 먼저 loading하도록 Class Loading 순서를 변경하는 것이다.

본 Webservice는 JAX-WS 2.0 이상을 사용한다.

TmaxSoft JEUS 6.0

전자정부 Webservice의 경우 내부적으로 CXF를 사용하지만 JEUS 6.0에 배포했을 경우 Server 모듈을 공개(publish)할 때 문제가 발생한다. 그 원인은 JEUS 6.0에 기본적으로 포함되어 있는 Web Services 관련 library와 전자정부 Webservice가 사용하는 library가 같지 않기 때문이다. 현재 아래와 같은 2가지 문제가 발견되었다.

- Publish Address 문제
Server 모듈을 publish할 때 EgovWebServiceServlet의 path에 대한 상대경로를 사용한다. Apache CXF가 사용하는 library의 경우, 이를 실제 주소로 변환해주지만, JEUS 6.0에 기본적으로 포함된 library는 그렇지 않기 때문에 IllegalArgumentException을 발생시킨다.
- Service Endpoint Interface 참조 문제

전자정부 WebService는 Integration 서비스 표준에 따라 Server 모듈의 Service Endpoint Interface와 구현 class를 동적으로 생성한다. 하지만 JEUS 6.0에 기본적으로 포함된 library의 경우, 이렇게 동적으로 생성된 class를 인식하지 못해서 Exception이 발생한다.

해결방법은 전자정부 WebService가 사용하는 library가 ClassLoader에서 먼저 loading되게 하는 것이다. JEUS 6.0은 jeus-web-dd.xml 설정을 통해서 WEB-INF/lib에 있는 library를 먼저 loading하도록 설정할 수 있다.

```
<?xml version="1.0" encoding="UTF-8"?>
<jeus-web-dd xmlns="http://www.tmaxsoft.com/xml/ns/jeus" version="6.0">
  <webinf-first>true</webinf-first>
</jeus-web-dd>
```

위 jeus-web-dd.xml 파일을 web.xml 파일이 존재하는 WEB-INF 폴더에 위치시킨다. 그리고, webinf-first를 true로 설정하는 경우, XML Parser에 대한 충돌이 발생한다. 충돌을 해결하기 위해서 아래 2개의 파일을 WEB-INF/lib에서 제거해야 한다.

- 'xml-apis-1.0.b2.jar' (또는 상위 버전)
- 'stax-api-1.0.1.jar' (또는 상위 버전)

JBoss

JBoss의 경우, 아래 jboss-web.xml 파일을 추가한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web>
  <class-loading java2ClassLoadingCompliance="false">
    <loader-repository>
      apache.cxf.archive=<WAR 파일명>
      <loader-repository-config>
        java2ParentDelegation=false
      </loader-repository-config>
    </loader-repository>
  </class-loading>
</jboss-web>
```

* <WAR 파일명>은 deploy하는 war 파일명을 확장자를 포함하여 기재한다.

WebLogic

WebLogic 9.2 버전은 J2EE 1.4까지만 지원하므로, JAX-WS 2.0을 지원하지 않는다. WebService를 WebLogic에서 사용하기 위해서는 JAX-WS 2.0 이상을 지원하는 10.x 이상을 사용해야 한다.

참고자료

- [Integration Service](#)
- <http://cxf.apache.org>

개요

Spring MVC를 통해 구현한 RESTful은 리소스에 대한 접근을 URI를 이용하며, HTTP의 PUT, GET, POST, DELETE 등과 같은 메소드의 의미를 그대로 사용하므로, 단순하게 접근할 수 있다.

설명

- Restful
 - 개요
 - 설명
 - web.xml 설정
 - Request Mapping
 - HTTP Method Conversion
 - HTTP Method Conversion
 - Views
 - 실제 사용 예
 - 참고자료

web.xml 설정

```
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.xml</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.json</url-pattern>
</servlet-mapping>

<filter>
  <filter-name>httpMethodFilter</filter-name>
  <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>httpMethodFilter</filter-name>
  <url-pattern>/springrest/*</url-pattern>
</filter-mapping>
```

자세한 설명은 아래에 있다.

Request Mapping

▪ 설정

REST 스타일의 URL은 '/cgr', '/cgr/CATEGORY-00000000001' 처럼 계층 구조로 사용가능하도록 설계되었다. 따라서 web.xml에 DispatcherServlet을 정의하고 매핑할 URL 패턴을 '/'로 지정해야한다. **DispatcherServlet URL 매핑** 샘플은 다음과 같다.

```
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>/springrest/*</url-pattern>
</servlet-mapping>
```

아래와 같은 방법으로도 **DispatcherServlet URL 매핑**을 사용 할 수 있다.

```
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.xml</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.json</url-pattern>
</servlet-mapping>
```

▪ 사용

Spring에서 제공하는 REST 지원 기능들은 모두 Spring MVC 기반으로 되어 있다. REST 방식으로 노출되는 서비스는 곧 Controller의 메소드이기 때문에 기존에 웹 어플리케이션을 개발하던 방식과 크게 다르지 않다.

Resource의 ID인 URI를 Controller 클래스나 메소드에 매핑하기 위해서는 **@RequestMapping**을 사용한다. @RequestMapping이 URI Template을 지원하기 때문에 아래 샘플코드와 같이 사용할 수 있다.

```
@Controller
@SessionAttributes(types=CategoryVO.class)
public class EgovCategoryController {
    //...
```

```
@RequestMapping(value="/springrest/cgr/{ctgryId}", method=RequestMethod.GET)
public String updtCategoryView(@PathVariable String ctgryId, Model model) throws Exception{
    // ...
}
```

모든 HTTP method 사용을 위해서 @RequestMapping에서 'method' 속성을 제공한다. 따라서, '/springrest/cgr/CATEGORY-00000000001'이라는 URI가 GET으로 요청이 들어올 경우 위의 updtCategoryView () 메소드가 매핑될 것이다.

▪ @PathVariable annotation추가

'/springrest/cgr/CATEGORY-00000000001'로 URI요청이 들어왔을 경우 @PathVariable을 사용하여 'ctgryId' 입력 인자로 바인딩 된다.

```
@RequestMapping(value="/springrest/cgr/{ctgryId}", method=RequestMethod.GET)
public String updtCategoryView(@PathVariable String ctgryId, Model model) throws Exception{
    // ...
}
```

HTTP Method Conversion

▪ 설정

브라우저 기반의 HTML에서는 GET, POST만 지원한다. 일반적으로 HTTP에서는 POST를 사용하고, hidden 타입의 입력값으로 HTTP METHOD를 지정하는 경우가 많다. 다음은 web.xml에 HiddenHttpMethodFilter를 정의한 모습이다.

```
<filter>
  <filter-name>httpMethodFilter</filter-name>
  <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>httpMethodFilter</filter-name>
  <url-pattern>/springrest/*</url-pattern>
</filter-mapping>
```

▪ 사용

web.xml에 HiddenHttpMethodFilter 설정을 추가하면, HTTP Method가 POST이고 _method라는 파라미터가 존재하는 경우 HTTP의 Method를 _method 값으로 바꾼다.

또한 Spring에서는 <form:form>에서 실제 HTTP Method를 지정하는 hidden 타입의 입력 필드를 자동으로 추가해 주기 때문에 훨씬 더 편리하게 사용할 수 있다.

```
<form:form method="delete">
  <input type="submit" value="Delete"/>
</form:form>
```

JSP에 위와 같이 작성하면, 내부적으로는 POST 방식으로 "_method=delete"가 전달되는 것이다.

샘플코드이다.

```
function fncSubmit(method){
    document.detailForm._method.value=method;
    document.detailForm.submit();
}

//..

<form:form name="detailForm" method="${method}">
  <a href="javascript:fncSubmit('delete');">삭제</a>
</form:form>
```

HTTP Method Conversion

Xml과 json 등 다른 view로 보여지는 것으로 spring에서는 ContentNegotiatingViewResolver를 제공한다. ContentNegotiatingViewResolver는 다른 View Resolver들과 반드시 함께 사용되어야 하므로 View Resolver 설정 시 반드시 order를 정의해야 한다. 당연히 ContentNegotiatingViewResolver가 가장 높은 우선순위(가장 작은숫자)를 가져야 한다. defaultView는 View를 찾지 못한 경우 디폴트 View로 사용된다.

```
<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
  <property name="defaultViews">
    <list>
      <bean class="org.springframework.web.servlet.view.json.MappingJacksonJsonView">
        <property name="prefixJson" value="false"/>
      </bean>
    </list>
  </property>
</bean>
```

Views

▪ MarshallingView

클라이언트에게 xml 응답을 돌려주기 위해 Spring OXM Marshaller를 사용한다. Spring oxm는 JAXB2, XMLBeans, JIBX, Castor등을 사용하여 Marshaller를 손쉽게 정의할 수 있게 해준다. Restful 예제에서는 JAXB2를 사용하였다. (OXM예제는 Castor사용)

```
<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
  <property name="mediaTypes">
    <map>
      <entry key="html" value="text/html" />
      <entry key="xml" value="application/xml" />
      <entry key="json" value="application/json" />
    </map>
  </property>
  <property name="order" value="0" />
</bean>

<bean name="cgr/egovCategoryRegister" class="org.springframework.web.servlet.view.xml.MarshallingView">
  <property name="marshaller" ref="marshaller" />
</bean>

<oxm:jaxb2-marshaller id="marshaller">
  <oxm:class-to-be-bound name="egovframework.rte.tex.cgr.service.CategoryVO" />
</oxm:jaxb2-marshaller>
```

▪ MappingJacksonJsonView

JSON으로 응답을 전달할 수 있는 View.

```
<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
  <property name="mediaTypes">
    <map>
      <entry key="html" value="text/html" />
      <entry key="xml" value="application/xml" />
      <entry key="json" value="application/json" />
    </map>
  </property>
  <property name="order" value="0" />
</bean>

<bean name="cgr/egovCategoryList"
class="org.springframework.web.servlet.view.json.MappingJacksonJsonView" />
```

실제 사용 예

▪ jsp

```
function fncSubmit(method) {
    document.detailForm._method.value=method;
    document.detailForm.submit();
}

//..

<form:form name="detailForm" method="${method}">
//..
</form:form>

//..
<a href="javascript:fncSubmit('post');">등록</a> //----- controller 2.
<a href="javascript:fncSubmit('put');">수정</a> //----- controller 3.
<a href="javascript:fncSubmit('delete');">삭제</a> //----- controller 4.
<a href="/springrest/cgr/{id}.xml">xml 보기</a> // ContentNegotiatingViewResolver설정
<a href="/springrest/cgr/{id}.json">json(defaultView) 보기</a> // ContentNegotiatingViewResolver설정
<a href="/springrest/cgr.html">목록</a> //----- controller 1.
<a href="/springrest/cgr.json">목록 (json)</a> // ContentNegotiatingViewResolver 설정
```

▪ controller

```
// 1. 목록
@RequestMapping(value="/springrest/cgr", method=RequestMethod.GET)
public String selectCategoryList(..) throws Exception {
    //..
}

// 2. 등록
@RequestMapping(value="/springrest/cgr", method = RequestMethod.POST, ..)
public String create(..) throws Exception {
    //..
}

// 3. 수정
@RequestMapping(value = "/springrest/cgr/{ctgryId}", method = RequestMethod.PUT, ..)
public String update(..) throws Exception {
    //..
}

// 4. 삭제
@RequestMapping(value = "/springrest/cgr/{ctgryId}", method=RequestMethod.DELETE)
```



```
public String deleteCategory(@PathVariable String ctgryId, SessionStatus status) throws Exception{  
    //..  
}
```

참고자료

-  RESTful 예제