

표준프레임워크 3.2기반 nexacroplatform 개발자 가이드

v 1.0



|  |  |  |  |
| --- | --- | --- | --- |
| **문서 개정이력표** | | | |
| **문서명** | | 표준프레임웍3.2 기반 nexacroplatform 개발자 가이드 | |
| 버전 | 날짜 | 내용 | 작성자 |
| 1.0 | 2015.03.24 | nexacro platform 개발자 가이드 작성 |  |

저작권 및 면책조항

본 문서는 기술상의 부정확함 또는 글자상의 오류를 포함하고 있으며, (주)투비소프트는 여기에 관한 어떠한 책임도 지지 않습니다.

본 문서의 내용은 정기적으로 변경 또는 추가 될 수 있으며, 그에 따른 변경본은 본 문서에 ‘New Edition’의 형태로 순차적으로 추가될 것입니다. (주)투비소프트는 사전 공지 없이 언제든지 본 문서에 기재된 제품 및 프로그램을 변경 또는 수정할 수 있습니다.

본 문서는 저작권법의 보호를 받으며, 어떠한 경우에도 (주)투비소프트의 사전승인 없이 임의로 무단 복사 또는 복제하여 배포할 수 없습니다.

목 차

[1. 개요 6](#_Toc414979744)

[1.1. 개발환경 구성요소 6](#_Toc414979745)

[2. eGovframe 3.2 – nexacro platform 실행환경 7](#_Toc414979746)

[2.1. eGovframe 3.2 – nexacroplatform 구성 7](#_Toc414979747)

[2.1.1. Architecture 8](#_Toc414979748)

[2.2. Presentation Layer 8](#_Toc414979749)

[2.2.1. Configuration Files 9](#_Toc414979750)

[2.2.2. Controller Class 17](#_Toc414979751)

[2.3. Business Layer 20](#_Toc414979752)

[2.3.1. Interface 21](#_Toc414979753)

[2.3.2. Implementation Class 22](#_Toc414979754)

[2.4. Persistence Layer 27](#_Toc414979755)

[2.4.1. DAO 27](#_Toc414979756)

[2.4.2. Configuration 28](#_Toc414979757)

[2.4.3. Mapping XML File 29](#_Toc414979758)

[2.4.4. Dao Class 31](#_Toc414979759)

[3. eGovframe 3.2 – mysql-5.6.21연동 32](#_Toc414979760)

[3.1. Query XML 작성 32](#_Toc414979761)

[3.2. Query Config파일에 쿼리 XML 추가하기 33](#_Toc414979762)

[4. nexacroplatform UI 구성요소 34](#_Toc414979763)

[4.1. nexacro platform 구성 파일 34](#_Toc414979764)

[4.2. nxui(html5) 프로젝트 카테고리 35](#_Toc414979765)

[4.3. 공통라이브러리구성 36](#_Toc414979766)

[4.3.1. Js Namespace 기법활용 36](#_Toc414979767)

[4.3.2. 모듈구성 36](#_Toc414979768)

[4.3.3. 구성요소(Eco Library + Ex Library) 37](#_Toc414979769)

[4.3.4. 작성방법 38](#_Toc414979770)

[4.3.5. 공통 xjs(comLib.xjs) 작성방법 38](#_Toc414979771)

[5. 코딩 규칙 39](#_Toc414979772)

[5.1. 명명규칙 39](#_Toc414979773)

[5.2. 주석 표기 방식 39](#_Toc414979774)

[6. nxui application project 41](#_Toc414979775)

[6.1. 소개 41](#_Toc414979776)

[6.2. 프로젝트 구성 41](#_Toc414979777)

[7. 실행방법 42](#_Toc414979778)

[8. nxui 기능사항 43](#_Toc414979779)

[8.1. 초기 로딩성능 개선사항 43](#_Toc414979780)

[8.1.1. iframe연동 로그인페이지 43](#_Toc414979781)

[8.1.2. 초기 로딩페이지 연동 43](#_Toc414979782)

[9. 공통기능사항 44](#_Toc414979783)

[9.1. 권한관리 44](#_Toc414979784)

[9.2. 디버깅기능제공 44](#_Toc414979785)

[9.3. 샘플기능 45](#_Toc414979786)

[9.4. 배포 46](#_Toc414979787)

[9.5. nexacro platform14 Deploy 기능 수행 적용 46](#_Toc414979788)

[10. 튜닝가이드(html5) 47](#_Toc414979789)

[10.1. nexacro14라이브러리 정제작업 47](#_Toc414979790)

[10.2. 서버환경적용사항 (Apach Tomcat) 47](#_Toc414979791)

[10.2.1. keep-alive 설정 47](#_Toc414979792)

[10.2.2. gZip적용 48](#_Toc414979793)

[10.2.3. 적용확인 48](#_Toc414979794)

[11. 측정 및 분석 49](#_Toc414979795)

[11.1. nexacro platform (index.html)로드분석 49](#_Toc414979796)

[11.1.1. Timeline 49](#_Toc414979797)

[11.2. 성능측정 50](#_Toc414979798)

[12. playnexacro(플레이넥사크로) 51](#_Toc414979799)

[13. 웹디버거 사용 51](#_Toc414979800)

# 개요

본 문서는 표준프레임웍(eGovframe3.2)와 nexacroplatform을 기반으로 하는 시스템 구축 시 개발자가 실제로 작업을 해야되는 내용에 대해 eGovframe framework의 각 Layer의 클래스를 작성하는 위주로 가이드를 제공함으로써 개발자가 일관성이 있는 개발을 하도록 지원하기 위해 작성되었다. nexacroplatform에 대한 지식은 어느 정도 갖추었거나 교육을 받은 것으로 한정하고 자세한 nexacroplatform 설명은 본 문서의 범주에서 제외했다.

개발 시 본 문서와 레퍼런스 사이트([www.nexacroplatform.co.kr](http://www.nexacroplatform.co.kr), www.egovframe.go.kr)를 통해 배포하는 소프트웨어를 활용하여 호환성의 문제가 발생하지 않도록 한다. 소프트웨어의 추가나 업그레이드가 있을 경우, 본 문서를 수정하여 추가 배포, 공지하도록 한다. 본 문서에는 개발환경이 구축되어 있다고 가정한다. (개발환경에 대한 내용은 eGovframe-nexacroplatform 개발환경구축가이드 문서를 참조)

매뉴얼을 기반으로 순서대로 따라해 보는 형식으로 구성되어 있다.

## 개발환경 구성요소

|  |  |  |  |
| --- | --- | --- | --- |
| 개발환경 구성요소 | | 개인 PC 환경 | |
| 환경설정 | Java Development Kit | JDK | 1.7.x Version 이상 |
| IDE | Eclipse | Eclipse (Kepler Service Release 2) 추천 |
| WAS | Tomcat | v7.0 |
| Frame work | EgovFrame | eGovFrame 3.2 |
| UI | nexacro platform | HTML5버전 |

[표 1‑1]

# eGovframe 3.2 – nexacro platform 실행환경

## eGovframe 3.2 – nexacroplatform 구성

nexacroplatform( )은 데이터셋이라는 고유한 형태를 사용해서 DB에 조회/입력/수정/삭제 동작을 수행하기 위해 Map형태로 구조화시킨 NexacroAdaptor와 공통 컨트롤러 클래스를 제공한다.

|  |  |
| --- | --- |
| 구성 | 용도 |
| **Controller (N**exacroController) | |  | | --- | | 비즈니스 서비스의 메소드를 호출하고 결과값을 화면으로 전달한다. | |
| **Service 인터페이스 (**NexacroService) | |  | | --- | | DataSetList, VariableList를 이용해 DB에서 데이터를 조회, 추가, 삭제, 수정 등을 할 수 있는 API를 제공한다. | |
| **Service 구현 클래스 (**NexacroServiceImpl**)** | |  | | --- | | NexacroService의 구현 클래스로 DataSet과 실행  하고자 하는 Query Id를 짝지은 후 NexacroDao의 메소드를 호출하고 Query실행 결과를 DataSetList에 추가한다. | |
| **Dao 클래스**  **(**NexacroDao) | 파라미터의 형태에 따라 조회,추가,삭제,수정 등을 할 수 있는 API를 제공한다. |
| **Dao 구현클래스**  **(SqlMapNexacroDAO)** | 파라미터의 형태에 따라 적절한 SQLMap의 메소드를 호출해 쿼리를 실행한다. |

[표 2‑2]

### Architecture

본 문서의 이해를 돕기 위하여 전자정부 표준프레임웍3.2 전체 소프트웨어 아키텍처를 그림으로 나타내면 다음과 같다.

아래 그림에서 노란색으로 표시된 부분이 넥사크로플랫폼 UIAdaptor를 이용해서 서비스 인터페이스를 하기 위해 작성한 부분이다.

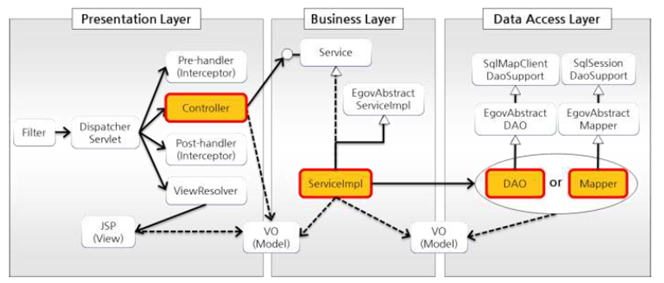
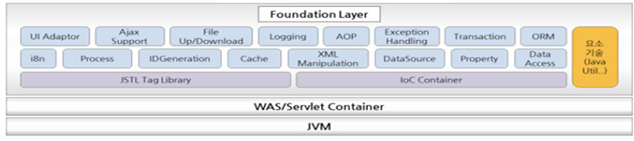
특히 구축하고자 하는 웹 Application이 전자정부 표준프레임웍과 상호호환성 검증을 받기 위해서는 아래 내용을 반드시 준수해야 한다.

- ServiceImpl클래스는 반드시 **EgovAbstractServiceImpl를 상속**받아서 구현해야 한다.

- (iBatis)의 경우 DAO클래스는 반드시 **EgovAbstractDAO를 상속**받아서 구현해야 한다.

- (MyBatis)의 경우Mapper 클래스는 반드시 **EgovAbstractMapper를 상속**받아서 구현해야 한다.

**예) 전자정부 표준프레임웍 아키텍쳐**

****

## Presentation Layer

Presentation Layer는 Client의 요청이나 입력 내용을 비즈니스 로직에 따라서 처리 후 처리된 결과를 Client에게 반환하는 역할을 수행한다.

### Configuration Files

****- web.xml****

C:\ eGovFrame-3.2 \workspace\nexacroplatform\webapps\WEB-INF (Eclipse Project Explorer 기준)

- Java EE 웹 어플리케이션은 반드시 WEB-INF 디렉토리 하위에 web.xml파일이 존재해야 한다.

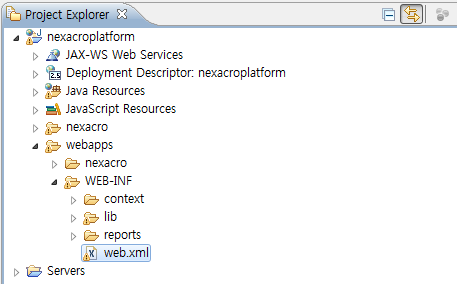
- tomcat의 실행환경에 대한 정보를 담당하는 '**환경설정**' 파일 이다.

- 각종 servlet의 설정과 servlet 매핑, 필터, 인코딩 등을 담당한다.

- web.xml은 톰캣에 있는 모든 web application의 기본설정을 정의한다.   
- web.xml은 각 application이 deploy될 때 각 application의 'WEB-INF/web.xml' deployment

descripter에 따라서 처리가 된다.

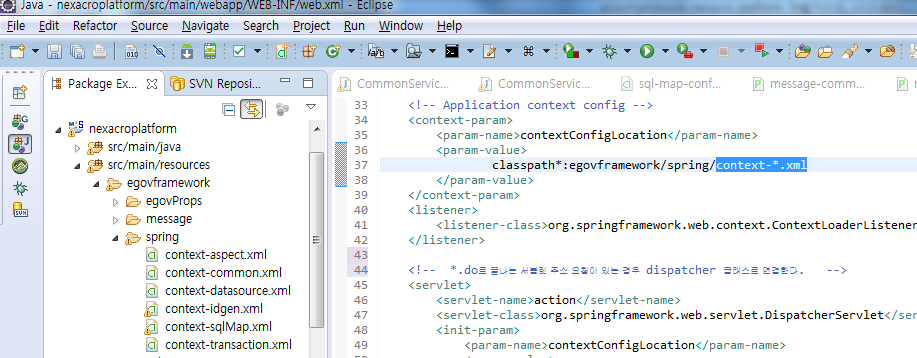
이 파일에 Front Controller 인 **DispatcherServlet** 을 등록한다.



|  |
| --- |
| <!—  서로 다른 DispatcherServlet에서 공통 빈을 사용 하고자 할 때에는 다음과 같이  context-param 과 ContextLoaderListener 을 이용한다.  contextConfigLocation: 설정파일의 위치 파라메터  -->  <!-- Application context config -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>  classpath\*:egovframework/spring/context-\*.xml  </param-value>  </context-param>  <listener>  <listener-class>  org.springframework.web.context.ContextLoaderListener  </listener-class>  </listener>  <!-- \*.do로 끝나는 서블릿 주소 요청이 있는 경우 dispatcher 클래스로 연결한다. -->  <!-- Servlet Mpaaing Config -->  <servlet>  <servlet-name>action</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>  /WEB-INF/config/egovframework/springmvc/dispatcher-servlet.xml,  </param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>action</servlet-name>  <url-pattern>\*.do</url-pattern>  </servlet-mapping>  <!-- Excel Import/Export Config -->  <servlet>  <servlet-name>XExportImport</servlet-name>  <servlet-class>com.nexacro.xeni.services.GridExportImportServlet</servlet-class>  </servlet>  <servlet>  <servlet-name>XImport</servlet-name>  <servlet-class>com.nexacro.xeni.services.GridImportServlet</servlet-class>  </servlet>    <servlet-mapping>  <servlet-name>XExportImport</servlet-name>  <url-pattern>/XExportImport</url-pattern>  </servlet-mapping>  <servlet-mapping>  <servlet-name>XImport</servlet-name>  <url-pattern>/XImport</url-pattern>  </servlet-mapping>    <context-param>  <param-name>export-path</param-name>  <param-value>/export</param-value>  </context-param>  <context-param>  <param-name>import-path</param-name>  <param-value>/import</param-value>  </context-param>  <context-param>  <param-name>monitor-cycle-time</param-name>  <param-value>30</param-value>  </context-param>  <context-param>  <param-name>file-storage-time</param-name>  <param-value>10</param-value>  </context-param> |

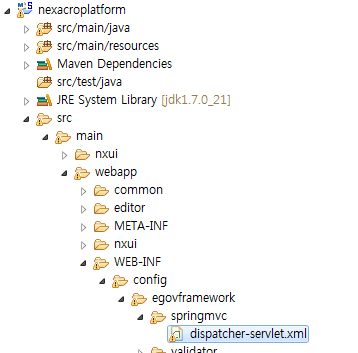
**-context-\*.xml**

Spring 설정 파일, 위 web.xml 설정에 따라서 listener에 의해서 어플리케이션이 시작될 시점에 메모리에 로드 한다.

****

\*-servlet.xml

\*-servlet.xml 에는 Controller 수행에 사용되는 bean정보를 설정하는 <bean> element가 있다.



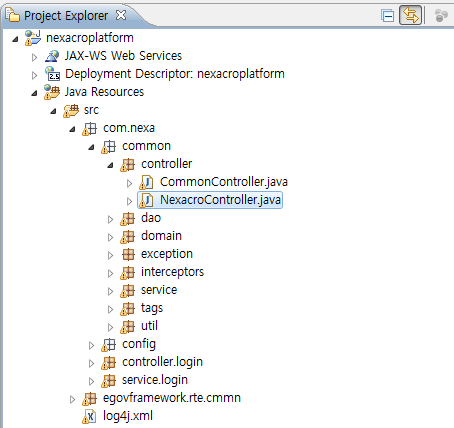
-dispatcher-servlet.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <!--  - DispatcherServlet application context for Application's web tier.  -->  <beans xmlns=*"http://www.springframework.org/schema/beans"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:p=*"http://www.springframework.org/schema/p"* xmlns:context=*"http://www.springframework.org/schema/context"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd"*>  <!--  The controllers are autodetected POJOs labeled with the @Controller annotation.  POJOs 레벨에서 @Controller로 선언된 컨트롤러들은 자동 감지되어 등록된다.  POJOs 레벨에서 @Service로 선언된 서비스들은 자동 감지되어 등록된다.  POJOs 레벨에서 @Repository로 선언된 저장공간들은 자동 감지되어 등록된다.  -->  <context:component-scan base-package=*"egovframework"*>  <context:include-filter type=*"annotation"* expression=*"org.springframework.stereotype.Controller"*/>  <context:exclude-filter type=*"annotation"* expression=*"org.springframework.stereotype.Service"*/>  <context:exclude-filter type=*"annotation"* expression=*"org.springframework.stereotype.Repository"*/>  </context:component-scan>  <!--  - The form-based controllers within this application provide @RequestMapping  - annotations at the type level for path mapping URLs and @RequestMapping  - at the method level for request type mappings (e.g., GET and POST).  - In contrast, ClinicController - which is not form-based - provides  - @RequestMapping only at the method level for path mapping URLs.  -  - DefaultAnnotationHandlerMapping is driven by these annotations and is  - enabled by default with Java 5+.  -->    <bean id=*"extAnnotationMapper"*  class=*"egovframework.rte.cmmn.ria.support.ExtAnnotationHandlerMapping"*>  <property name=*"order"* value=*"0"* />  <property name=*"rule"*>  <value>select</value>  </property>  <property name=*"urls"*>  <list>  <value>/nexacroController.do</value>  </list>  </property>  <property name=*"interceptors"*>  <list>  <ref bean=*"riaHandler"* />  </list>  </property>  </bean>  <bean id=*"riaHandler"* class=*"egovframework.rte.cmmn.ria.support.RiaInterceptor"*>  <property name=*"uiDTO"*>  <list>  <bean class=*"egovframework.rte.cmmn.ria.xplatform.map.XPlatformMapDTO"* />  </list>  </property>  </bean>      <bean id=*"localeChangeInterceptor"*  class=*"egovframework.rte.cmmn.ria.support.interceptor.LocaleChangeInterceptor"*>  <property name=*"paramName"* value=*"language"* />  </bean>  <bean id=*"annotationMapper"*  class=*"org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping"*>  <property name=*"order"* value=*"1"* />  <property name=*"interceptors"*>  <list>  <ref bean=*"localeChangeInterceptor"* />  </list>  </property>  </bean>    <!--  - This bean processes annotated handler methods, applying Application-specific PropertyEditors  - for request parameter binding. It overrides the default AnnotationMethodHandlerAdapter.  -->  <bean class=*"org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"*>  <property name=*"webBindingInitializer"*>  <bean class=*"egovframework.rte.cmmn.web.EgovBindingInitializer"*/>  </property>  <property name=*"customArgumentResolvers"*>  <list>  <bean class=*"egovframework.rte.cmmn.ria.support.CustomRiaArgumentResolver"*>  <property name=*"uiAdaptor"*>  <ref bean=*"nexacroMapAdaptor"* />  </property>  </bean>  </list>  </property>  </bean>    <bean name=*"nexacroMapAdaptor"* class=*"egovframework.rte.cmmn.ria.xplatform.map.NexacroAdaptorImp"* />  <!--  - This bean resolves specific types of exceptions to corresponding logical  - view names for error views. The default behaviour of DispatcherServlet  - is to propagate all exceptions to the servlet container: this will happen  - here with all other types of exceptions.  -->  <bean class=*"org.springframework.web.servlet.handler.SimpleMappingExceptionResolver"*>  <property name=*"defaultErrorView"* value=*"egovframework/cmmn/egovError"*/>  <property name=*"exceptionMappings"*>  <props>  <prop key=*"org.springframework.dao.DataAccessException"*>cmmn/dataAccessFailure</prop>  <prop key=*"org.springframework.transaction.TransactionException"*>cmmn/dataAccessFailure</prop>  <prop key=*"egovframework.rte.fdl.cmmn.exception.EgovBizException"*>cmmn/egovBizException</prop>  </props>  </property>  </bean>    <!--  - This bean configures the 'prefix' and 'suffix' properties of  - InternalResourceViewResolver, which resolves logical view names  - returned by Controllers. For example, a logical view name of "vets"  - will be mapped to "/WEB-INF/jsp/vets.jsp".  -->  <bean class=*"org.springframework.web.servlet.view.BeanNameViewResolver"*  p:order=*"0"* />    <bean id=*"nexacroMapView"* class=*"egovframework.rte.cmmn.ria.nexacro.map.NexacroMapView"* />    <!--  - Message source for this context, loaded from localized "messages\_xx" files.  - Could also reside in the root application context, as it is generic,  - but is currently just used within Application's web tier.  -->  <bean id=*"messageSource"* class=*"org.springframework.context.support.ResourceBundleMessageSource"*  p:basename=*"messages"*/>  </beans> |

### Controller Class

MVC에서 C에 해당하는 컨트롤러는 사용자의 요청을 받아서 어플리케이션에 정의된 적절한 Service를 수행한 후, 그 결과를 다시 View를 통해 사용자에게 보여줄 수 있는 Model 데이터로 변환하는 역할을 담당한다. Spring에서는 이러한 컨트롤러를 특정 API에 종속되지 않고 사용자가 자유롭게 작성할 수 있는 추상적인 구현 방법을 제공하고 있다.

Spring 2.5에서부터 @RequestMapping, @RequestParam, @ModelAttribute 등을 이용한Annotation 기반의 컨트롤러 개발 방식을 소개했다. Annotation을 사용하여 SpringMVC기반의 컨트롤러를 작성하면, 특정 인터페이스를 상속받거나 특정 클래스를 상속받지 않아도 된다. 또한 Servlet API와도 독립적으로 작성할 수 있다는 장점이 있다.



다음 예제는 Annotation 기반으로 선언된 NexacroController 의 예제이다.

|  |
| --- |
| package com.nexa.common.controller;  import java.lang.reflect.Method;  import java.util.Map;  import javax.annotation.Resource;  import org.apache.commons.logging.Log;  import org.apache.commons.logging.LogFactory;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.servlet.ModelAndView;  **import** egovframework.com.nexa.common.service.NexacroService;  **import** egovframework.rte.cmmn.ria.nexacro.NexacroConstant;  **import** egovframework.rte.cmmn.ria.nexacro.map.DataSetMap;  **import** egovframework.rte.cmmn.ria.nexacro.map.NexacroMapDTO;  **import** egovframework.rte.cmmn.ria.nexacro.vo.NexacroObjectDTO;  @Controller("nexacroController")  public class NexacroController {    protected Log logger = LogFactory.getLog(this.getClass());  @Resource(name="nexacroService")  NexacroService nexacroService;      @RequestMapping("/nexacroController.do")  public ModelAndView operate(NexacroMapDTO xpDto, Model model) throws Exception {    ModelAndView mav = new ModelAndView("nexacroMapView");    try  {  DataSetMap tranInfo = xpDto.getTranInfoMap();  Map <String, Object> invar = xpDto.getInVariableMap();  Map <String, DataSetMap> inDataset = xpDto.getInDataSetMap();  Map <String, Object> outVar = xpDto.getOutVariableMap();  Map <String, DataSetMap> outDataset = xpDto.getOutDataSetMap();    Object bean = nexacroService;    Method method = getMethod(bean, (String)inVar.get("method"));  try{  method.invoke(bean, new Object[] {tranInfo, inVar, inDataset, outVar, outDataset});  } catch (Exception e){  Throwable te = e.getCause();  logger.debug("Can not invoke a dispatch method name", te);  throw new Exception(te.getMessage(), te);  }  mav.addObject(NexacroConstant.OUT\_VARIABLES\_ATT\_NAME, xpDto.getOutVariableMap());  mav.addObject(NexacroConstant.OUT\_DATASET\_ATT\_NAME, xpDto.getOutDataSetMap());    mav.addObject(NexacroConstant.ERROR\_CODE, "0");  mav.addObject(NexacroConstant.ERROR\_MSG, "");    }  catch ( Exception e )  {  e.printStackTrace();  mav.addObject(NexacroConstant.ERROR\_CODE, "-1");  mav.addObject(NexacroConstant.ERROR\_MSG, e.toString());  }  return mav;  }    private Method getMethod(Object bean, String methodName) throws Exception{  Method[] methods = bean.getClass().getMethods();  for(int i = 0 ; i < methods.length ; i++) {  if( methods[i].getName().equals(methodName)) {  return methods[i];  }  }  throw new Exception("Cann't find " + methodName + ".");  }  } |

위 소스에서 사용된 Annotation syntax와 역할은 다음과 같다.

|  |  |
| --- | --- |
| Annotation Syntax | 역할 |
| @Controller | 특정 클래스에 @Controller annotation을 적용하면 다른 클래스를 상속받거나 Servlet API를 사용하지 않아도 해당 클래스가 Controller 역할을 수행 |
| @RequestMapping | Controller 클래스나 메소드가 특정 HTTP Request URL을 처리하도록 매핑하기 위해서 사용. |
| @Service | 특정 클래스에 @Service annotation을 적용하면 다른 클래스를 상속 받거나 Servlet API를 사용하지 않아도 해당 클래스가 Service 역활을 수행한다. |

[표 2‑3]

## Business Layer

일반적으로 business layer를 구성하는 서비스는 아래 그림과 같이 서비스 인터페이스, 서비스 구현 클래스, 그리고 데이터 접근 로직을 구현한 Dao 클래스, 그리고 서비스와 참조 관계를 가지고 있는 클래스, 서비스의 속성을 정의한 annotation 으로 이루어져 있다.

**Bussiness Layer**

**Interface**

**Implematation**

**DAO (Interface)**

**SQLMap\*DAO**

Annotation 기반의 Depandency설정

Mapping XML File  
(mapping-\*.xml)

**Dependency Class1**

**Dependency Class2**

**Dependency ClassN**

**Model**

**Controller**

**lass n**

**AbstractDao**

: Annotation 기반 의존성 설정 : 개발자가 개발해야 할 부분

업무 처리 로직은 비즈니스 영역에서 처리되어야 하는데(예: 트랜잭션이 필요한 여러 테이블에 대한 작업 또는 복잡한 계산 로직 등) 반드시 Interface 와 Implementation 클래스로 작성하여야 한다. 클래스간 참조관계에 대한 환경설정은 annotation 을 통해 정의한다.(XML 설정을 통해서도 클래스간 참조 관계에 대한 설정이 가능하나 annotation 사용을 원칙으로 한다.)

### Interface

해당 서비스가 제공할 기능을 정의한다.

[Interface 예제]

|  |
| --- |
| package egovframework.com.nexa.common.service;  import java.util.Map;  import egovframework.rte.cmmn.ria.nexacro.map.DataSetMap;  @SuppressWarnings("unchecked")  public interface NexacroService {  void get(  DataSetMap tranInfo,  Map<String, Object> inVar, Map<String, DataSetMap> inDataset,  Map<String, Object> outVar, Map<String, DataSetMap> outDataset) throws Exception;    void getList(  DataSetMap tranInfo,  Map<String, Object> inVar, Map<String, DataSetMap> inDataset,  Map<String, Object> outVar, Map<String, DataSetMap> outDataset) throws Exception;    int save(  DataSetMap tranInfo,  Map<String, Object> inVar, Map<String, DataSetMap> inDataset,  Map<String, Object> outVar, Map<String, DataSetMap> outDataset) throws Exception;    void doModifyVOService(  DataSetMap tranInfo,  Map<String, Object> inVar, Map<String, DataSetMap> inDataset,  Map<String, Object> outVar, Map<String, DataSetMap> outDataset) throws Exception;  } |

Interface 의 일반 적인 작성 방법은 위와 같다.

이처럼 **인터페이스와 구현 클래스를 분리해 놓은 이유는 서비스 이용자가 구현 클래스의 구체적인 비즈니스 구현 로직을 알 필요가 없이 인터페이스 클래스에서 제시하고 있는 메소드만 파악하고 사용할 수 있도록 하기 위해서이며**, eGovframe 의 근간인 Spring Framework 은 기본적으로 Interface / Implementation 구조를 통해 Framework 의 기능들을 적용하기 때문이다.

### Implementation Class

Implementation Class에서는 interface에서 정의한 method 를 구현한다.

NexacroServiceImpl의 예제는 다음과 같다.

|  |
| --- |
| package egovframework.com.nexa.common.service;  import java.util.Iterator;  import java.util.List;  import java.util.Map;  import javax.annotation.Resource;  import org.springframework.stereotype.Service;  import com.nexacro.xapi.data.DataSet;  **import** egovframework.com.nexa.common.dao.BoardDAO;  **import** egovframework.com.nexa.common.dao.NexacroDAO;  **import** egovframework.com.nexa.common.dao.SampleDAO;  **import** egovframework.com.nexa.common.domain.Board;  **import** egovframework.com.nexa.common.domain.SampleDefaultVO;  **import** egovframework.rte.cmmn.ria.nexacro.NexacroConstant;  **import** egovframework.rte.cmmn.ria.nexacro.map.DataSetMap;  **import** egovframework.rte.fdl.cmmn.EgovAbstractServiceImpl;  **// @Service : 서비스 명을 지정**  **// @Transactional : 예외 발생시 롤백 하도록 지정 (기본 트랜잭션인 Required 적용)**  @Service("nexacroService")  @SuppressWarnings("unchecked")  public class NexacroServiceImpl extends **EgovAbstractServiceImpl** implements NexacroService {    @Resource(name="sqlMapBoardDAO")  BoardDAO boardDao;    @Resource(name="sqlMapNexacroDAO")  NexacroDAO nexacroDao;  public void get(DataSetMap tranInfo,  Map<String, Object> inVar, Map<String, DataSetMap> inDataset,  Map<String, Object> outVar, Map<String, DataSetMap> outDataset)  throws Exception {  getList(tranInfo, inVar, inDataset, outVar, outDataset);  }  public void getList(DataSetMap tranInfo,  Map<String, Object> inVar, Map<String, DataSetMap> inDataset,  Map<String, Object> outVar, Map<String, DataSetMap> outDataset)  throws Exception {  String[] arySqlId = ((String)inVar.get("sqlId")).split(" ");  for (int i = 0; i < arySqlId.length; i++) {  DataSetMap inDsMap = (DataSetMap)inDataset.get("input" + (i+1));  Map inMap = null;  if (inDsMap != null)  inMap = inDsMap.get(0);  List <Map> records;  if (inMap != null) {  records = nexacroDao.getList(arySqlId[i], inMap);  } else {  records = nexacroDao.getList(arySqlId[i], inVar);  }  DataSetMap outDsMap = new DataSetMap();  outDsMap.setRowMaps(records);  outDataset.put("output" + (i+1), outDsMap);  }  }  public void getListPaging(DataSetMap tranInfo,  Map<String, Object> inVar, Map<String, DataSetMap> inDataset,  Map<String, Object> outVar, Map<String, DataSetMap> outDataset)  throws Exception {    String[] arySqlId = ((String)inVar.get("sqlId")).split(" ");    for (int i = 0; i < arySqlId.length; i++) {  DataSetMap inDsMap = (DataSetMap)inDataset.get("input" + (i+1));  Map inMap = null;  if (inDsMap != null)  inMap = inDsMap.get(0);  List <Map> records;  if (inMap != null) {  records = nexacroDao.getList(arySqlId[i], inMap);  } else {  records = nexacroDao.getList(arySqlId[i], inVar);  }  DataSetMap outDsMap = new DataSetMap();  outDsMap.setRowMaps(records);  outDataset.put("output" + (i+1), outDsMap);  }  }  public int save(DataSetMap tranInfo,  Map<String, Object> inVar, Map<String, DataSetMap> inDataset,  Map<String, Object> outVar, Map<String, DataSetMap> outDataset)  throws Exception {  String sqlId = null;  String forceFlag = null;  String[] arySqlId = ((String)inVar.get("sqlId")).split(" ");  String[] forceSqlFlag = ((String)inVar.get("forceSqlFlag")).split(" ");  int iRow = 0;  for (int i = 0; i < arySqlId.length; i++) {  DataSetMap list = (DataSetMap) inDataset.get("input" + (i+1));  sqlId = arySqlId[i];  for (int x = 0; x < list.size(); x++) {  Map map = list.get(x);  //DataSet에 Parameter를 추가  Iterator<String> keys = inVar.keySet().iterator();  while( keys.hasNext() ){  String key = keys.next();  map.put(key, inVar.get(key));  }  if(forceSqlFlag[i].toUpperCase().equals("Y")){  iRow += nexacroDao.update(sqlId, map);  }else{  int rowType = ((Integer) map.get(NexacroConstant.DATASET\_ROW\_TYPE)).intValue();  forceFlag = null;  if ( rowType == DataSet.ROW\_TYPE\_INSERTED ){  if (forceSqlFlag[i].toUpperCase().equals("N"))  forceFlag = "\_I";  nexacroDao.insert(sqlId + forceFlag, map);  }else if (rowType == DataSet.ROW\_TYPE\_UPDATED){  if (forceSqlFlag[i].toUpperCase().equals("N"))  forceFlag = "\_U";  iRow += nexacroDao.update(sqlId + forceFlag, map);  }else if (rowType == DataSet.ROW\_TYPE\_DELETED){  if (forceSqlFlag[i].equals("N"))  forceFlag = "\_D";  iRow += nexacroDao.delete(sqlId + forceFlag, map);  }  }  }  }  return iRow;  }  // Vo 형태로 파라메터 전달  public void getListBoard(DataSetMap tranInfo,  Map<String, Object> inVar, Map<String, DataSetMap> inDataset,  Map<String, Object> outVar, Map<String, DataSetMap> outDataset)  throws Exception {  String[] arySqlId = ((String)inVar.get("sqlId")).split(" ");  for (int i = 0; i < arySqlId.length; i++) {  DataSetMap inDsMap = (DataSetMap)inDataset.get("input" + (i+1));  Map inMap = null;  if (inDsMap != null)  inMap = inDsMap.get(0);  List <Map> records;  if (inMap != null) {  records = boardDao.selectBoardList( inMap);  } else {  records = boardDao.selectBoardList( inVar);  }  DataSetMap outDsMap = new DataSetMap();  outDsMap.setRowMaps(records);  outDataset.put("output" + (i+1), outDsMap);  }  }  public void doModifyVOService(DataSetMap tranInfo,  Map<String, Object> inVar, Map<String, DataSetMap> inDataset,  Map<String, Object> outVar, Map<String, DataSetMap> outDataset)  throws Exception {  //DataSetMap inDsMap = (DataSetMap)inDataset.get("input1");  DataSetMap list = (DataSetMap) inDataset.get("input1");    Board board = new Board();  for (int x = 0; x < list.size(); x++) {  Map map = list.get(x);  //DataSet에 Parameter를 추가  Iterator<String> keys = inVar.keySet().iterator();  while( keys.hasNext() ){  String key = keys.next();  map.put(key, inVar.get(key));  }  ////////////////////////////////////////////////////////  board.setTitle( (String)map.get("TITLE"));  board.setReg\_id((String)map.get("REG\_ID"));  board.setContents((String)map.get("CONTENTS"));  board.setPost\_id((String)map.get("POST\_ID"));  board.setCommunity\_id((String)map.get("COMMUNITY\_ID"));  ////////////////////////////////////////////////////////  int rowType = ((Integer) map.get(NexacroConstant.DATASET\_ROW\_TYPE)).intValue();  if (rowType == DataSet.ROW\_TYPE\_INSERTED){  boardDao.insertBoard(board);  }else if (rowType == DataSet.ROW\_TYPE\_UPDATED){  boardDao.updateBoard(board);  }else if (rowType == DataSet.ROW\_TYPE\_DELETED){  boardDao.deleteBoard(board);  }  }  }  } |

서비스 구현 클래스에서는 서비스 인터페이스에서 정의한 모든 메소드를 구현해야

한다. 위 소스에서 사용된 Annotation syntax와 역할은 다음과 같다.

|  |  |
| --- | --- |
| Annotation Syntax | 역할 |
| @Service | 비즈니스 로직을 처리하는 클래스를 정의하는데 사용한다. |
| @Transactional | Transaction 관리 대상 클래스 또는 method를 식별한다. @Transactional Annotation은 method 위에 나타낼 수도 있으며 상세 속성을 지시 할 수 있다. 상세 속성은 eGovframe 커뮤니티(http://www.eGovframejava.org)를 참조한다. |

[표 2-4]

## Persistence Layer

Persistence Layer는 DBMS에 접근하여 데이터를 처리하는 역할을 담당한다. iBatis에서 제공하는 SQLMap를 사용한다. Persistence Framework이란 JDBC boilerplate 코드를 없애고 DB 접근 작업을 간소화 / 정형화한 프레임워크를 의미한다.

### DAO

DAO 서비스는 쿼리문이나 객체의 입력만으로 DB 데이터 조작을 가능하게 하는 api를 제공하는 인터페이스이다. 그리고 SqlMap 서비스는 JDBC를 이용한 데이터 액세스 수행 부분을 추상화 함으로써 간편한 데이터 액세스 방법을 제공하고, JDBC 사용 시 발생할 수 있는 공통적인 에러를 줄여준다. 본 문서에서는 Mapping XML File 기반으로 한SqlMap 사용 방법에 대해서 필수적인 부분을 다루기로 하며, 그 외 다른 이용방법에 대한 내용은 온라인 매뉴얼을 참조 한다.0

### Configuration

속성 정의의 복잡성을 제거하고 간편하게 MySQLMap의 속성을 정의할 수 있도록 지원하기 위해 query namespace를 제공한다.

[context-sqlMap.xml]

|  |
| --- |
| <beans  xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:tx="http://www.springframework.org/schema/tx"  xmlns:aop="http://www.springframework.org/schema/aop"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop-2.5.xsd  http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">  <aop:aspectj-autoproxy proxy-target-class="true"/>  <!-- iBatis SQLMAP Config -->  <bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">  <property name="configLocation"  value="classpath:/egovframework/sqlmap/rte/sql-map-config.xml"/>  <property name="dataSource" ref="dataSource"/>  </bean> |

****auto-config 속성****

* dbType : altibase, db2, hsqldb, mysql, oracle, others 중에서 선택 가능하며, 정의된 dbType에 의해 pagingSQLGenerator와 lobHandler Bean의 구현체가 결정된다.
* id : QueryService Bean에 대해 별도의 식별자를 부여하고자 하는 경우
* jdbcTemplate-ref : 기본적으로 'jdbcTemplate'이라는 이름의 Bean을 참조하여 실행되는데 QueryService가 제공하는 org.eGovframe.query.impl.jdbc.PagingJdbcTemplate 구현체가 아닌 다른 template을 활용하고자 하는 경우 사용
* sqlLoader-ref : 기본 적용되는 'sqlLoader' Bean의 클래스패스 상의 sql/query/\*.xml 위치가 아닌 다른 매핑 XML 파일의 위치 및 기타 속성을 달리 적용하고자 하는 경우 사용
* dataSource-ref : 기본적으로 'dataSource'라는 이름의 Bean을 사용하는데 다
* 른 DataSource Bean을 참조하여 QueryService를 실행하고자 하는 경우
* 위의 내용 외의 쿼리서비스의 설정 정보에 대한 내용은 EGovframe 매뉴얼을 참조 한다.

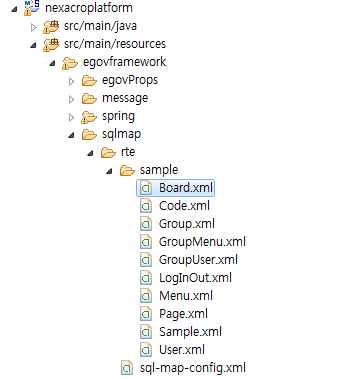
### Mapping XML File

SqlMap 서비스 초기화 시, SqlMap 서비스는 속성 정의 파일에 정의되어 있는 mapping XML file을 로드 한다. 그리고 사용자 요청 시 mapping 정보를 기반으로 query id를 이용해 실행하고자 하는 쿼리문을 찾아 실행한다. Mapping XML file은 <queryservice>와 </queryservice>내에 크게 <table-mapping>과 <queries>로 구성된다. <queries>는 필수 요소이다. 기본적인 Mapping XML File의 작성법과 활용법은 개발 템플릿상의 예제 파일 및 매뉴얼을 참고하고, 본 문서에서는 Mapping XML File의 샘플 예제와 파일 위치 설정에 대해서 설명하도록 한다.

다음의 예제는 Mapping XML File 의 예제(board.xml) 이다. Query id 는 전체 Mapping XML File 상에서 중복이 전혀 없어야 한다는 점에 유의하여야 한다.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE sqlMap PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN" "http://www.ibatis.com/dtd/sql-map-2.dtd">  <!--  /\*\*  \*  \* WARNING: Auto create  \* Template File Version : 1.0  \* Create : Tue Jul 07 11:48:58 KST 2009  \*  \*\*/  -->  <sqlMap namespace="Group">  <!-- code -->  <select id="selectGroupList" parameterClass="java.util.HashMap" resultClass="java.util.HashMap">  SELECT  A.GROUP\_ID AS GROUP\_ID,  A.GROUP\_NM AS GROUP\_NM,  A.GROUP\_DESC AS GROUP\_DESC,  A.USE\_FG AS USE\_FG,  A.CREATE\_USER AS CREATE\_USER,  A.CREATE\_DT AS CREATE\_DT,  A.UPDATE\_USER AS UPDATE\_USER,  A.UPDATE\_DT AS UPDATE\_DT  FROM TB\_GROUP A  </select>    <statement id="modyfyGroup\_I" parameterClass="java.util.Map" resultClass="java.util.Map" remapResults="true">  <![CDATA[  insert into TB\_GROUP (GROUP\_ID, GROUP\_NM, GROUP\_DESC, USE\_FG) values(#GROUP\_ID#,#GROUP\_NM#,#GROUP\_DESC#,#USE\_FG#);  ]]>  </statement>    <statement id="modyfyGroup\_U" parameterClass="java.util.Map" resultClass="java.util.Map" remapResults="true">  <![CDATA[  UPDATE TB\_GROUP  SET GROUP\_NM=#GROUP\_NM#  , GROUP\_DESC=#GROUP\_DESC#  , USE\_FG=#USE\_FG#  WHERE GROUP\_ID=#GROUP\_ID#  ]]>  </statement>    <statement id="modyfyGroup\_D" parameterClass="java.util.Map" resultClass="java.util.Map" remapResults="true">  <![CDATA[  DELETE FROM TB\_GROUP  WHERE GROUP\_ID=#GROUP\_ID#  ]]>  </statement>  </sqlMap> |

Mapping XML File의 위치는 관리와 배포를 위해 src/main/resource/sqlmap/rte/sample/ 디렉토리에서 일괄적으로 관리한다. 하위 업무 별로 디렉토리 추가는 가능하다.



### Dao Class

Dao 클래스는 DBMS에 접근해서 데이터를 처리하는 클래스이며 iBatis 를 활용하여 데이터 접근 로직을 처리한다. 이러한 Dao 클래스를 간편하게 개발할 수 있도록 하기 위해 **모든 Dao 클래스는** 전자정부표준프레임웍에서 제공하는 **EgovAbstractDAO 클래스를 상속받아 구현**한다.

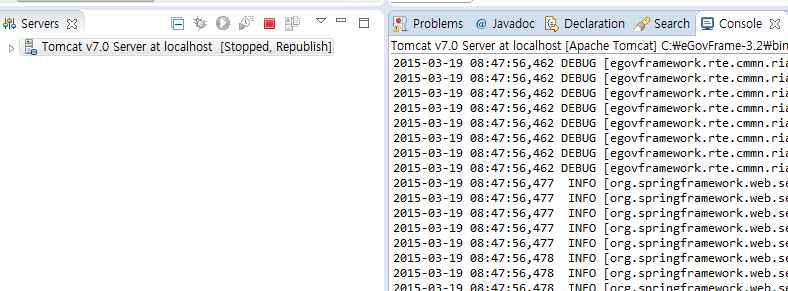
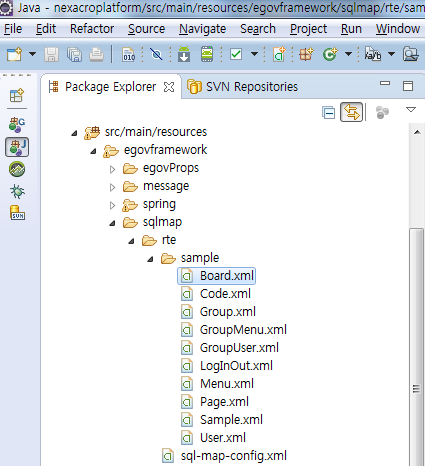
# eGovframe 3.2 – mysql-5.6.21연동

## Query XML 작성

DB연결과 사용 table은 이미 생성되어 있다고 가정한다. Eclipse에서 Board.xml을 열고 아래 query구문을 추가하고 파일을 저장한다.

* 쿼리파일 위치:

**C:\eGovFrame-.2\workspace\nexacroplatform\src\main\resources\egovframework\sqlmap\rte\sample**

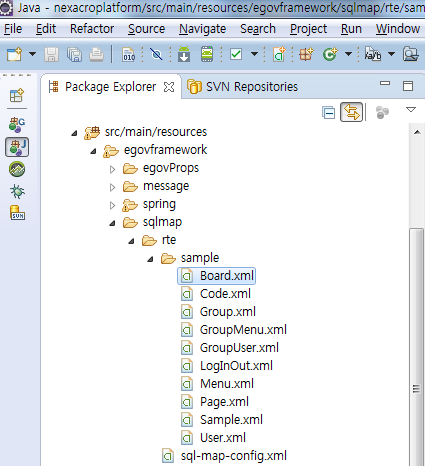


**Servers탭에서 설정된 Tomcat을 선택하고 우측마우스에서 Restart를 선택해서 서버를 재 기동시킨다.**

|  |
| --- |
| <!-- code -->  <!-- board -->  <select id="selectBoardList" parameterClass="java.util.HashMap"  resultClass="java.util.HashMap">  SELECT  A.TITLE AS TITLE,  A.REG\_ID AS REG\_ID,  A.REG\_DATE AS REG\_DATE,  A.POST\_ID AS POST\_ID,  A.CONTENTS AS CONTENTS,  A.COMMUNITY\_ID AS COMMUNITY\_ID  FROM XP\_BOARD A  <isNotEmpty property="TITLE">  WHERE TITLE LIKE "%$TITLE$%"  </isNotEmpty>  </select> |

## Query Config파일에 쿼리 XML 추가하기

업무가 추가되면 위와 같이 sql-map-config.xml 파일을 열고 추가로 생성한 쿼리 xml을 추가한다.

****

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE sqlMapConfig PUBLIC "-//iBATIS.com//DTD SQL Map Config 2.0//EN"  "http://www.ibatis.com/dtd/sql-map-config-2.dtd">  <sqlMapConfig>  <sqlMap resource="egovframework/sqlmap/rte/sample/LogInOut.xml" />  <sqlMap resource="egovframework/sqlmap/rte/sample/User.xml" />  <sqlMap resource="egovframework/sqlmap/rte/sample/Code.xml" />  <sqlMap resource="egovframework/sqlmap/rte/sample/Board.xml" />  <sqlMap resource="egovframework/sqlmap/rte/sample/Group.xml" />  <sqlMap resource="egovframework/sqlmap/rte/sample/GroupUser.xml" />  <sqlMap resource="egovframework/sqlmap/rte/sample/Menu.xml" />  <sqlMap resource="egovframework/sqlmap/rte/sample/Page.xml" />  <sqlMap resource="egovframework/sqlmap/rte/sample/GroupMenu.xml" />  <sqlMap resource="egovframework/sqlmap/rte/sample/Sample.xml" />  </sqlMapConfig> |

# nexacroplatform UI 구성요소

## nexacro platform 구성 파일

nexacro platform UI를 구성하는 여러 파일들의 종류와 용도를 설명한다.

|  |  |  |
| --- | --- | --- |
| 파일의 종류 | 파일명 | 용도 |
| 화면 폼 정의 | \*.xfdl | |  |  | | --- | --- | | - 응용프로그램이 사용하는 화면 폼을 정의합니다.   |  | | --- | | - 컴포넌트들의 화면 배치정보를 담고 있습니다. - 이벤트 핸들링방식의 UI 로직을 담고 있습니다. | | |
| 응용프로그램 정의 | \*.xadl | |  | | --- | | - 응용프로그램을 정의합니다.  - 응용프로그램 실행을 위한 환경정보를 담고 있습니다.  - 폼들의 배치정보를 담고 있습니다.  - 응용프로그램 UI모델 정보를 담고 있습니다.  - UI모델에는 2개의 방식(MDI,SDI)이 있습니다. | |
| 응용프로그램 자원의 정의 | |  | | --- | | default\_tpedef.xml | | |  | | --- | | - 응용프로그램이 사용하는 자원을 정의합니다.  - 응용프로그램이 사용하는 컴포넌트들을 정의합니다.  - 응용프로그램이 사용하는 데이터소스를 정의합니다.  - 응용프로그램이 사용하는 모듈들의 배포정보를 정의합니다. | |

[표 4‑1]

## nxui(html5) 프로젝트 카테고리

넥사크로 UI소스는 nxui(html5,rutime),nxmobile(모바일) 프로젝트 네이밍을 정해 진행중에 있습니다.

기본 공통 카테고리는 nexacro14 library에 module화하여 최초 로딩시 메모리에 정제된 라이브러리를 씀으로서 성능을 최대로 끌어올리는데 목적이 있습니다. 생태계조성팀에서 구축한 Eco Library리에 jsNamespace를 표준으로 정했으며 메인프레임 공통에서 사용하는 라이브러리는 Ex(External) Libary라고 명명규칙을 정했다.

comLib는 화면단에 xjs파일로 구성되며 화면과 연결고리를 만들어주는 공통 calback단 함수나 form event성 파일을 처리하게끔 구성한다.

주석은 jsDoc 규칙에 맞게 작성을 해야하며 api배포는 jsdoc문서로 배포를 한다.

## 공통라이브러리구성

### Js Namespace 기법활용

javascript namespace는 많은 프로그래밍 언어에서 대규모 프로그램을 제작할 때 다른 개체 나 전역 네임 스페이스의 변수와의 충돌을 방지하기 위해 사용하는 기술입니다.

또한 고유하게 식별 할 수 있고, 쉽게 관리 그룹으로 응용 프로그램의 기능 블록을 구성하는데

용이합니다.

javascript namespace 기법에서 함수 명칭에 Prefix을 정의하는 방법이 있지만, 이것은 사이트 상황에 따라 변경해야 요소가 많아서 사용하지 않습니다.

향후 넥사크로에 공통라이브러리에 표준은 playnexacro에 생태계조성팀에서 구현한 Eco Library에 기반인 Object Literal Notation 방식을 적용한 namespace기법을 활용하여 이번 프로젝트에 적용했으며

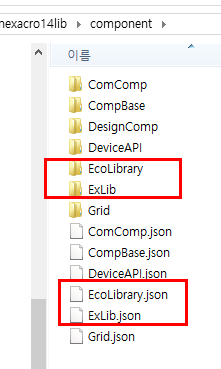
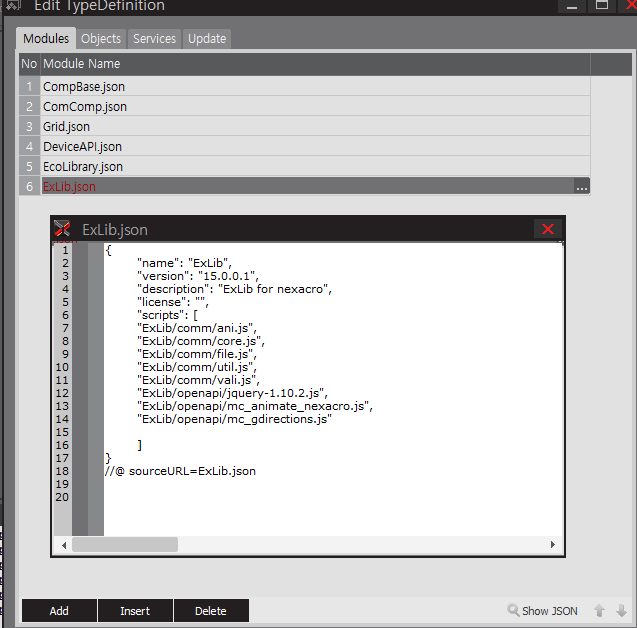
공통 메인프레임에서 사용하는 라이브러리에 명명은 ExLib라 정하여 작업을 진행하였습니다.

### 모듈구성

추후 유지보수를위해 Eco라이브러리와 공통메인프레임에서 사용하는 라이브러리를 분류했으며

넥사크로플랫폼은 여러분들도 아시겠지만 xplatform과 구조상 모듈이 별도 분리가되어 사용할수 있게끔 되어있으며 모듈에 json으로 탑재 js파일을 최초 메모리에 올려 넥사크로 로딩후 사용자는 최적에 성능을 느낄수 있도록 구성하였습니다. (Html5 기준)

* Eco Library : 기본 util 라이브러리는 Eco Library를 사용
* Ex Library : 메인프레임 공통 라이브러리

라이브러리에 작성은 Atomic Functionality 제공하기 위한 기본(Basic) 라이브러리

함수들은 사이트 의존성 없이 구현한다.

예) Eco.**string**. startsWith, Eco.**number**. bytesToSize

**Ex.**core.tran(this,sSvcID, sController, sInDatasets, sOutDatasets, sArgs);

* 참조사이트 [playnexacro]

Eco Library 다운로드: <http://www.playnexacro.com>

### 구성요소(Eco Library + Ex Library)

| 구분 | 내용 | 비고 |
| --- | --- | --- |
| Eco | 변수 값 형태(Type) 체크 함수들 |  |
| 값이 다른 형태(Type)로 Convert가능한지 체크 함수들  이미 nexacroplatform에서 기본적으로 제공되는 함수들은 제외  (isAlpha, isAlphaNumeric, isNumeric, isSpace, isUpper) |  |
| 빈값 체크하는 함수 - isEmpty  Null, undefined 이면 당연히 Empty라고 판단한다.  Null, undefined 이외에는 변수 값 Type 별로 Empty 처리한다. |  |
| 복제 하는 함수 – clone |  |
| 조건 제어 함수들 – decode, iif |  |
| Eco.Color | 색상이 명도, 채도 값에 의해 변경되는 색을 구하는 함수들  색상으로 데이터 시각화 처리할 때 유효한 함수 |  |
| Eco.date | 날짜 및 Date 형태(Type) 값을 처리하는 각종 Util 함수들 |  |
| Eco.Logger | "fatal", "error", "warn", "info", "debug" 레벨(level) 기준으로 메시지 표시하는 기능  로그 처리시 메시지 Format:  시간(yyyy-MM-dd hh:mm:ss.sss ) [Log Level] 함수명 : 메시지  추가 옵션 ( callStack, dump , elapsedTime, filter) |  |
| Eco.number | 숫자 및 Number 형태(Type) 값을 처리하는 각종 Util 함수들 |  |
| Eco.string | 문자열(String) 을 처리하는 각종 Util 함수들 |  |
| Eco.object | Simple object 즉 {} 로 선언된 값을 처리하는 각종 Util 함수들. |  |
| Eco.array | array 즉 [] 로 선언된 값을 처리하는 각종 Util 함수들. |  |
| Eco.XComp | XPLATFORM Component 관련된 처리 함수들(사이트 의존성이 없는 것들)  Query 검색 기능  Get/set Properties  Contain, lookup  위치 및 크기 관련 Util  style 관련 Util |  |
| Ex.core | 공통메인프레임에서 사용하는 함수를 제공  Transaction,공통팝업,공통 formOnload에서 사용하는 setting함수 네비게이션처리 이벤트핸들러,form 리사이징,엑셀임포트,익스포트 기타.. |  |
| Ex.file | Html5 이미지리사이즈 파일업로드 및 기본 파일업로드 |  |
| Ex.util | 메인프레임에서사용하는 utility 함수 제공 |  |
| Ex.vali | 공통 validation 함수 |  |
| Ex.ani | 3rd party opensource animation (jquery ,css3animation) |  |

### 작성방법

동적 Namespace에 따른 주석처리

* Library의 Namespace는 declare()를 통해서 동적으로 생성된다.
* 따라서 JSDoc에서 처리할 수 있도록 다음과 같이 @namespaced와 @memberOf Tag를 표기한다.

|  |
| --- |
| /\*\*  \* @namespace  \* @name Eco.date  \*/  Namespace.declare("Eco.date", {  /\*\*  \* 해당연도의 1월 1일부터 날짜까지의 일 수 반환.  \* **@param** {date} date Date Object.  \* **@return** {number} 평년은 0~364 반환. 윤년은 0~365 반환  \* @memberOf Eco.date  \*/  getDayOfYear: function(date)  {  var start = new Date(date.getFullYear(), 0, 0);  return this.getDiffDate(date, start) - 1;  }, |

Parameter의 Data Type 표현

* parameter의 타입은 중괄호( “{ }” )안에 표시한다.
* 소문자로 시작하는 camelCase를 사용한다.
* function의 parameter타입을 선언시에는 @param 태그를 사용한다.

### 공통 xjs(comLib.xjs) 작성방법

화면단과 연결고리로 xjs에서 공통 함수를 기술하며 콜벡관련된 함수들이나 event 성 함수들을 정의

함수명명규칙은 gfn\_로 처리

|  |
| --- |
| /\*\*  \* 이 함수가 먼저 수행되고 사용자지정Callback함수가 수행된다  \* @param {string} svcID  \* @param {string} 에러코드  \* @param {string} 에러 메시지  \* @return N/A  \* @example  \* @memberOf comLib  \*/  this.\_gfn\_callback = function(o, errorCode, errorMsg)  {  application.set\_usewaitcursor(false,true); //커서 비활성화.  var svcId = o.id;  var strMsg = errorMsg.length > 50 ? errorMsg.substr(0,50) + "..." : errorMsg;  next ………  } |

# 코딩 규칙

Library 구현시 코딩 규칙에 대한 설명이다.( Eco Library readme 문서에서 발췌)

## 명명규칙

대소문자 구분

| 대상 | 표기 | 예 |
| --- | --- | --- |
| Function | camel Case사용.  (backColor) | removeStr() |
| 변수 | var age = 10;  var userInfo = {}; |
| 상수 | 대문자 +”\_(underscore) “ + 대문자 | var A\_B = 1; |
| 생성자 | Pascal case 사용.  (BackColor) | var data = new HashMap();  var btn = new Button(); |
| 파일명 | Nexacro.xjs |

[표 5‑1]

단•복수 구분(복수시 Array개념 포함)

| 대상 | 예 |
| --- | --- |
| 명사 단•복수 구분 | var key, var keys |
| get, set Method 단•복수 구분 | getKey(), getKeys() |

[표 5‑2]

Variable, Parameter, Argument에 접두어를 사용하지 않는다.

※ 접두어를 사용하지 않으므로 명칭을 보고 판단할 수 있도록 명명을 한다.(아래 그림 참조)

※ Return Type은 주석 및 API Reference에 명시한다.(아래 그림 참조)

예) string, number(Decimal, Integer), boolean, array, object, variant

## 주석 표기 방식

Tag 목록

| 대상 | 예 |
| --- | --- |
| @constant | 상수임을 표시한다. |
| @constructor | 생성자임을 표시한다. |
| @name | JsDoc Toolkit에 주어진 이름을 사용하도록 표시한다. |
| @namespace | namespace를 표시한다. |
| @memberOf | 기술한 자원의 member임을 표시한다. |
| @private | private임을 표시한다. |
| @param | function의 parameter를 표시한다. |
| @return | function의 리턴값을 표시한다. |
| @require | 필요한 자원을 표시한다(의존성 표시). |
| @see | 연관된 자원에 link를 건다. |

[표 5‑3]

※ 자세한 사항은 아래 URL 참고

<https://code.google.com/p/jsdoc-toolkit/wiki/TagReference>

※ Table Of Contents를 표시하려면 @memberof**!**를 사용한다. @memberof! <global>

(강제 membership 태그)

# nxui application project

## 소개

nxui(html5) : nxui(html5)프로젝트는 넥사크로 플랫폼 SDI프레임 구성으로 실제웹페지에 가깝게 기능 구현을하였으며 전자정부프레임워크3.2와 연동하여 내부 정제작업에 필요한 여건을 고려하여 구성하였으며 인터넷이 안되는 환경에서도 확인할수 있게끔 로컬 DB로 구성이 되어있습니다.

HTML5 지원하는 모든브라우저를 지원하며 제약사항으로는 인터넷익스플로러는 **IE10이상** 브라우저에서만 지원합니다.

## 프로젝트 구성

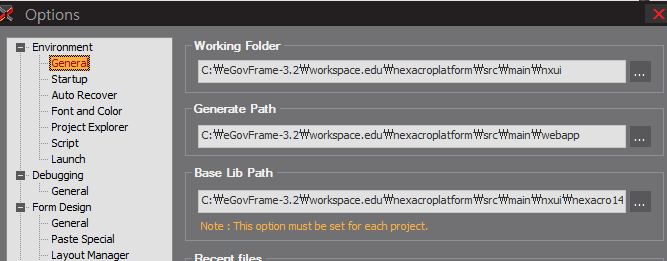
| 이미지 | 구분 | 설명 |
| --- | --- | --- |
|  | nexacro14lib.component | nexacro14lib.component |
| nexacro14lib.framework | nexacro14lib framework |
| nexacro14lib.resources | nexacro14lib.resources |
| web | 별도 web 페이지(login.html,loading.html) |
| frame | 메인프레임 폼 |
| lib | 공통 xjs파일 |
| extComp | 유저컴퍼넌트 |
| sample | 개발샘플 |
| admin | 권한관리화면 |
|  | CompBase | Nexacro14 lib 엔진 |
| ComComp |
| Grid |
| DeviceAPI |
| EcoLibrary | EcoLibrary |
| ExLibrary | 공통 |

# 실행방법

1. eGovFrame-3.2.7zip을 c:\에 **7zip** 압축프로그램을 사용하여 푼다
2. 넥사크로스튜디오 옵션 설정

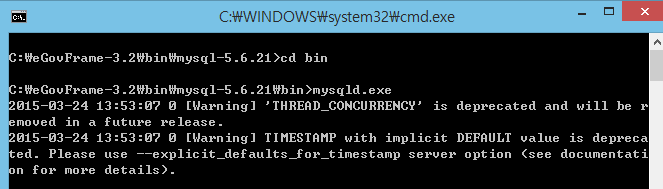
Generate Path : C:\eGovFrame-3.2\workspace.edu\nexacroplatform\src\main\webapp

Base Lib Path : C:\eGovFrame-3.2\workspace.edu\nexacroplatform\src\main\nxui\nexacro14lib

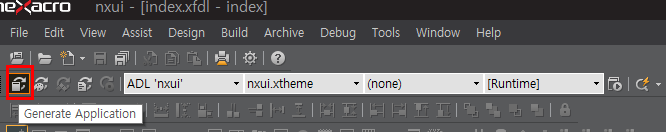


1. 로컬 db 구동

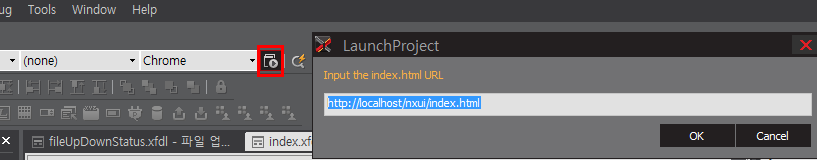
로컬Db 실행경로 : C:\eGovFrame-3.2\bin\mysql-5.6.21\ startup.bat



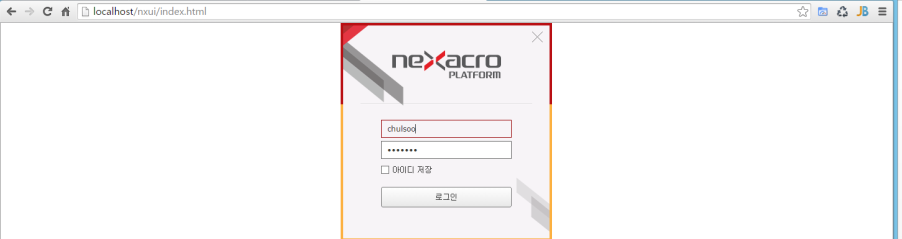
1. generate

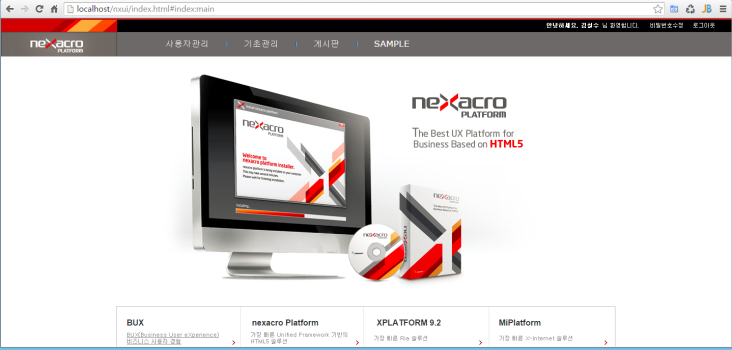


1. 런처 실행



1. 로그인 ID/PW 입력 (chulsoo/test123)





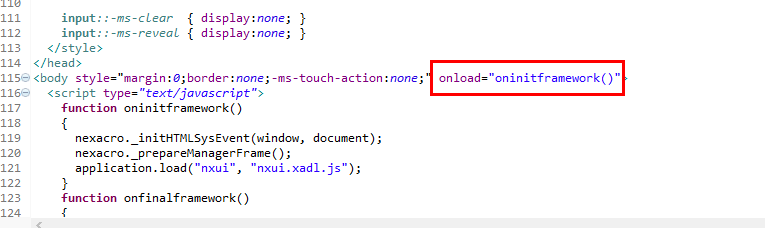
# nxui 기능사항

## 초기 로딩성능 개선사항

### iframe연동 로그인페이지

-초기로딩 속도를 개선하기위해 nexacro14Lib러리가 웹페이지에 올라오는동안 해당 사용자에게 보여주며 로그인처리시 프로그레스바를 보여준다.

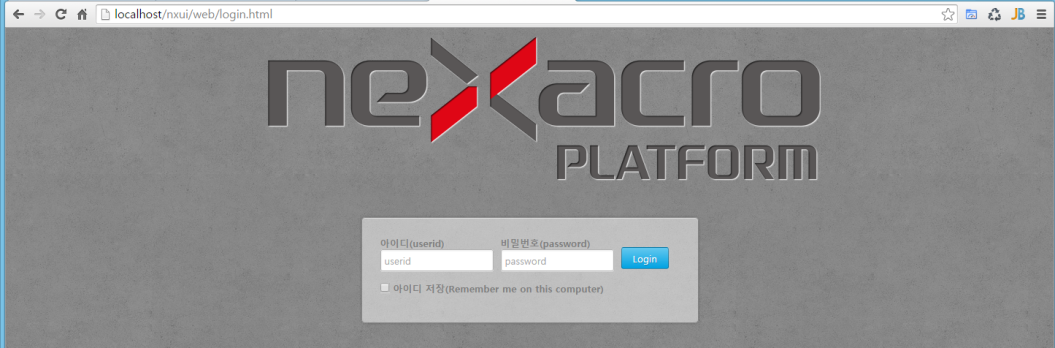
1. **index.html페이지에 onload=”oninitframewor()” 를 삭제**



삭제

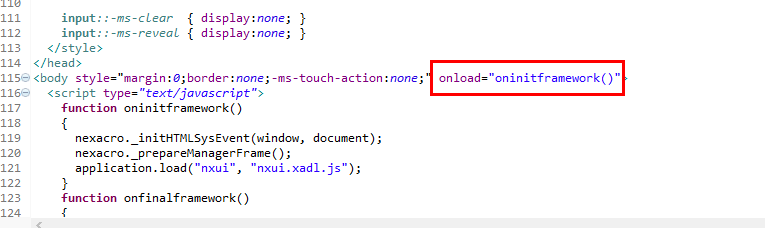
1. **브라우저에 url 입력**

URL : <http://localhost/nxui/web/login.html>



### 초기 로딩페이지 연동

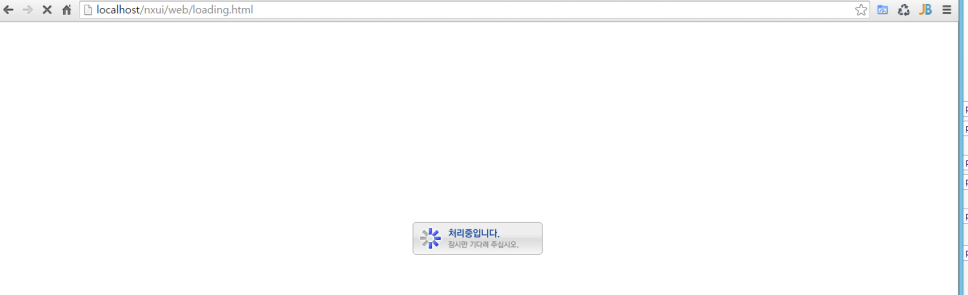
1. 1**index.html페이지에 onload=”oninitframewor()” 를 삭제**



삭제

1. **브라우저에 url 입력**

URL : <http://localhost/nxui/web/loading.html>



# 공통기능사항

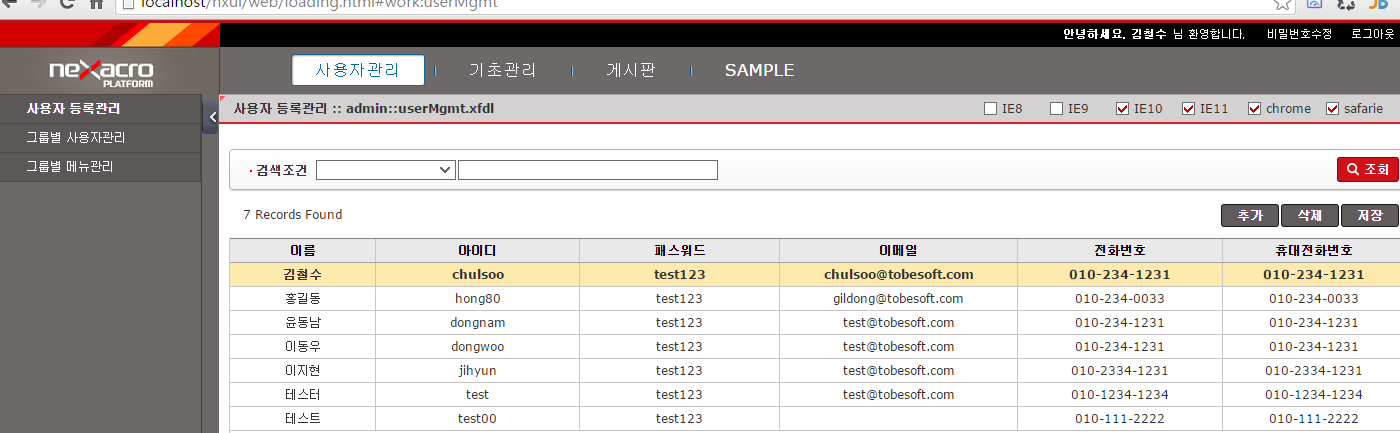
## 권한관리

기본 권한관리 템플릿에 대한 패턴을 제공하여 넥사크로에서 구현된 탬플릿유형을 확인해볼수 있다

* 리스트

| 화면명 | 업무단위 | 파일명 |
| --- | --- | --- |
| 공통코드관리 | admin | codeMgmt |
| 그룹별메뉴관리 | admin | groupMenuMgmt |
| 그룹관리 | admin | groupMgmt |
| 그룹별사용자관리 | admin | groupMgmt |
| 메뉴등록관리 | admin | menuMgmt |
| 화면등록관리 | admin | pageMgmt |
| 사용자등록관리 | admin | userMgmt |

* preview

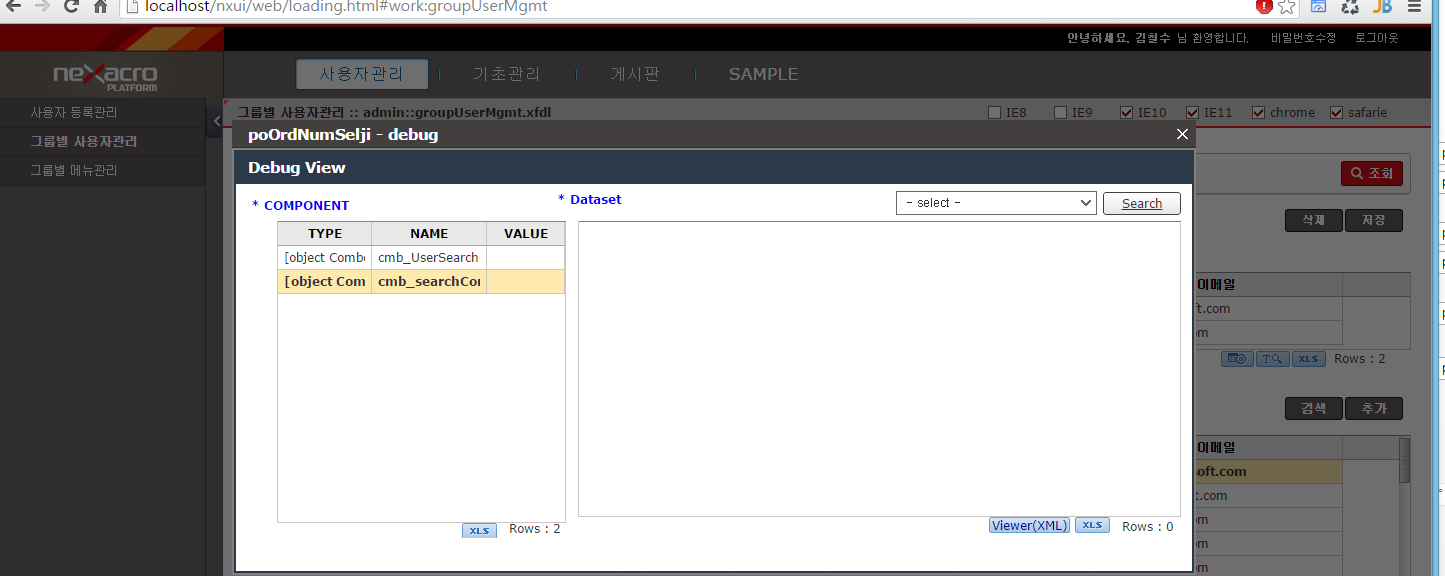


## 디버깅기능제공

개발자 디버깅 기능 제공

프로젝트에서 단축키 (ctl + alt + m)

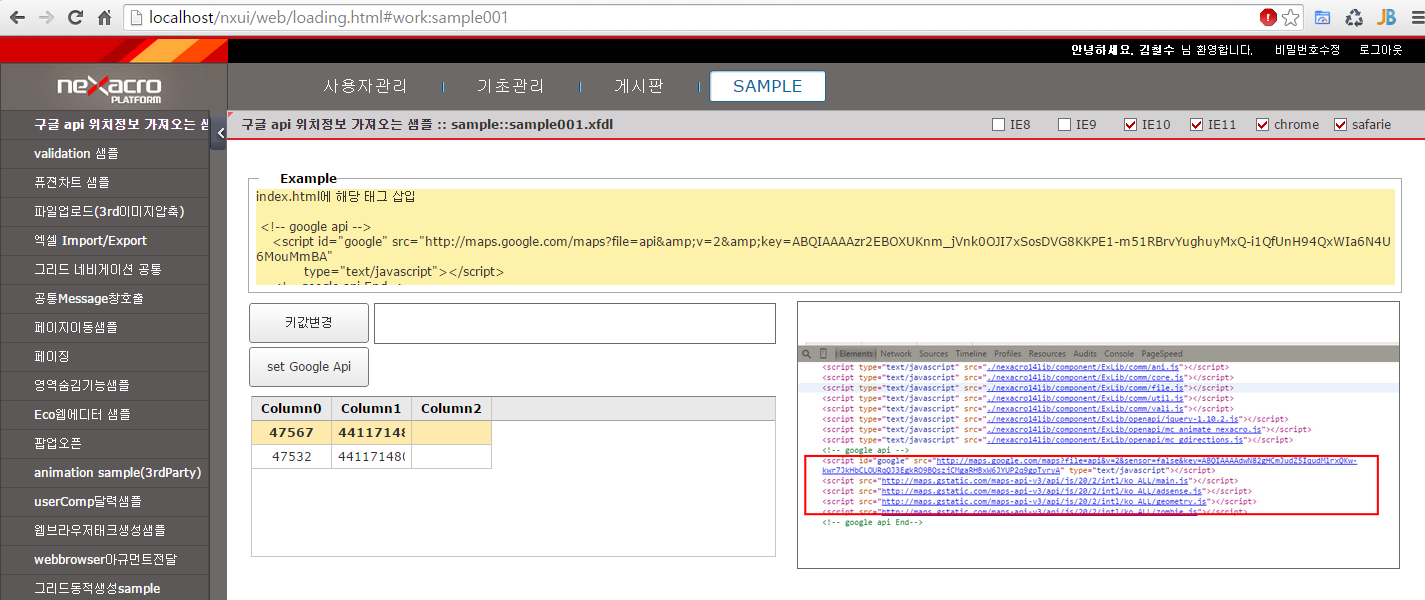
* preview



## 샘플기능

* 목록

| 화면명 | 업무단위 | 파일명 |
| --- | --- | --- |
| 구글api 위치정보가져오는샘플 | sample | sample001 |
| 페이징처리 | sample | sample002 |
| validation 샘플 | sample | sample003 |
| 영영기능숨김기능샘플 | sample | sample004 |
| 퓨전챠트연동샘플 | sample | sample005 |
| Eco 웹에디터 샘플 | sample | sample006 |
| 기본파일업/다운로드 | sample | sample007 |
| html5 fileReader태그사용 파일 및이미지압축전송 업로드 | sample | sample007\_1 |
| 공통팝업 | sample | sample008 |
| 엑셀 import/export | sample | sample009 |
| animation sample(3rd party) open source(jquery,css3animation) | sample | sample010 |
| 그리드 네비게이션 공통 | sample | sample011 |
| 달력컴퍼넌트(유저컴퍼넌트) | sample | sample012 |
| 공통 message(alert,confim) | sample | sample013 |
| 웹브라우저태그생성샘플 | sample | sample014 |
| 그리드 동적 생성샘플 | sample | sample015 |
| webbrowser 아규먼트 전달샘플 | sample | sample016 |
| 페이지 이동 샘플 | sample | sample017 |
| 브라우저 javascript hashmap(뒤로가기 앞으로가기) | sample | sample018 |



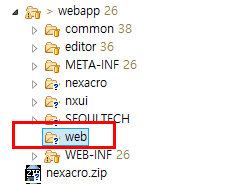
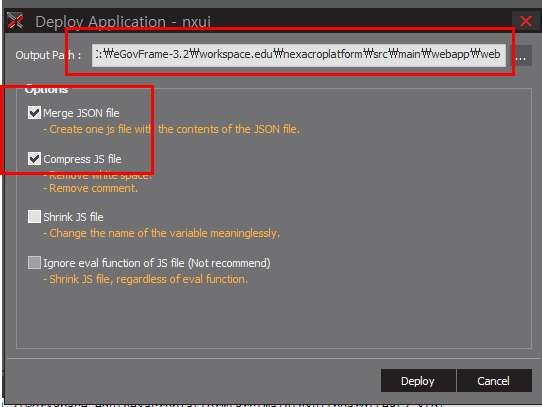
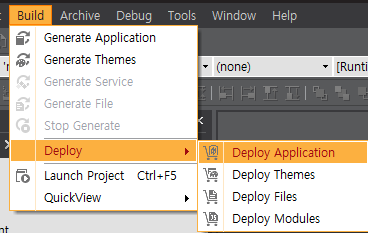
## 배포

## nexacro platform14 Deploy 기능 수행 적용

Deploy기능은 json파일을 js(자바스크립트로) 머지시켜 jsMean 형태로 만들어주며 용량을 최소화 시켜줘 성능을 최적화 시켜줍니다. 별도 Deploy 해줄 폴더는 만들어 사용하셔야합니다.

성능 개선을 위해 기본 개발은 generate로 개발하며 실제 배포는 넥사크로플랫폼14에 Deploy 기능을 사용하여 배포합니다.

1. 웹어플리케이션배포할 폴더에 web이라는 폴더를 만들어줍니다.

   
소스비교

| **generate 소스** | **deploy소스** |
| --- | --- |
|  |  |

# 튜닝가이드(html5)

## nexacro14라이브러리 정제작업

nexacr14lib러리에 엔진목록중 html5만으로 개발하신다면 런타임과 device에 사용하는 js파일을 삭제하면 초기로딩 성능향상을 기대할수 있습니다. 해당 목록은 html5로 제작된 사이트일경우 제거할 라이브러리 목록 입니다.

* nexacro14lib 제거목록

| 폴더명 | 제거파일명 |
| --- | --- |
| **ComComp(포함되어있는 json파일에 스크립트도 삭제해줘야함)** | nexacro14lib/component/ComComp/MediaPlayer.js  nexacro14lib/component/ComComp/Map.js  nexacro14lib/component/ComComp/Sketch.js |
| **CompBase(포함되어있는 json파일에 스크립트도 삭제해줘야함)** | nexacro14lib/component/CompBase/Step.js  nexacro14lib/component/CompBase/Element\_Runtime.js |
| **framework(별도 json파일에 스크립트도 삭제해줘야함)** | nexacro14lib/framework/Platform\_Runtime.js  nexacro14lib/framework/SystemBase\_Runtime.js  nexacro14lib/framework/Device\_iOS.js  nexacro14lib/framework/Device\_Android.js |
| **DeviceAPI** | nexacro14lib/DeviceAPI/SQLite.js",  nexacro14lib/DeviceAPI/SQLite\_Runtime.js",  nexacro14lib/DeviceAPI/DeviceObjs\_Runtime.js",  nexacro14lib/DeviceAPI/DeviceObjs.js” |
| **nexacro14lib/framework/Framework.json**  **목록에서 삭제** | SystemBase\_Runtime.js",  "Platform\_Runtime.js" |

## 서버환경적용사항 (Apach Tomcat)

### keep-alive 설정

특정 한 프로세스가 특정 사용자의 지속적인 요청 작업을 계속해서 처리하도록 함.

keepAliveTimeout값이 채워지기 전에 다시 요청이 온다면 다시 keepAliveTimeout은 다시 0부터 cont를 수행

* 사이트참조

<http://mcpaint.tistory.com/141>

<http://blog.naver.com/noorol/140199393456>

### gZip적용

**대부분 gzip이 어떤건지 아시리라 생각이 들지만 아래에 참조 url 내용 읽어보심 좋을 듯 합니다.**

gZip란?

gzip은 파일 압축에 쓰이는 응용 소프트웨어이다. gzip은 GNU zip의 준말이며, 초기 유닉스 시스템에 쓰이던 압축 프로그램을 대체하기 위한 자유 소프트웨어이다. gzip은 Jean-loup Gailly와 마크 애들러가 만들었다. 버전 0.1은 1992년 10월 31일에 처음 공개되었으며, 버전 1.0이 1993년 2월에 뒤따라 나왔다. 오픈BSD의 gzip 버전은 더 오래된 압축 프로그램을 기반으로 하고 있으며, 오픈BSD 3.4에 추가되었다

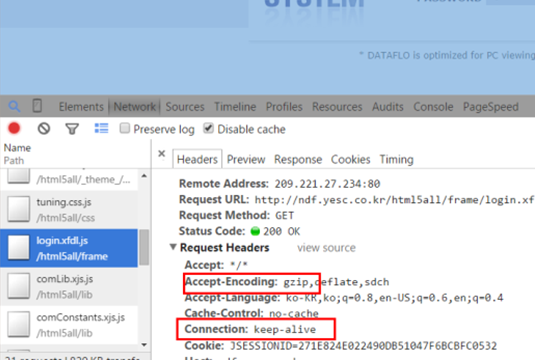
* 사이트참조

<http://pat.im/863>

<http://blog.naver.com/noorol/140199393456>

### 적용확인

크롬브라우저 단축키(f12) / network

. 

# 측정 및 분석

## nexacro platform (index.html)로드분석

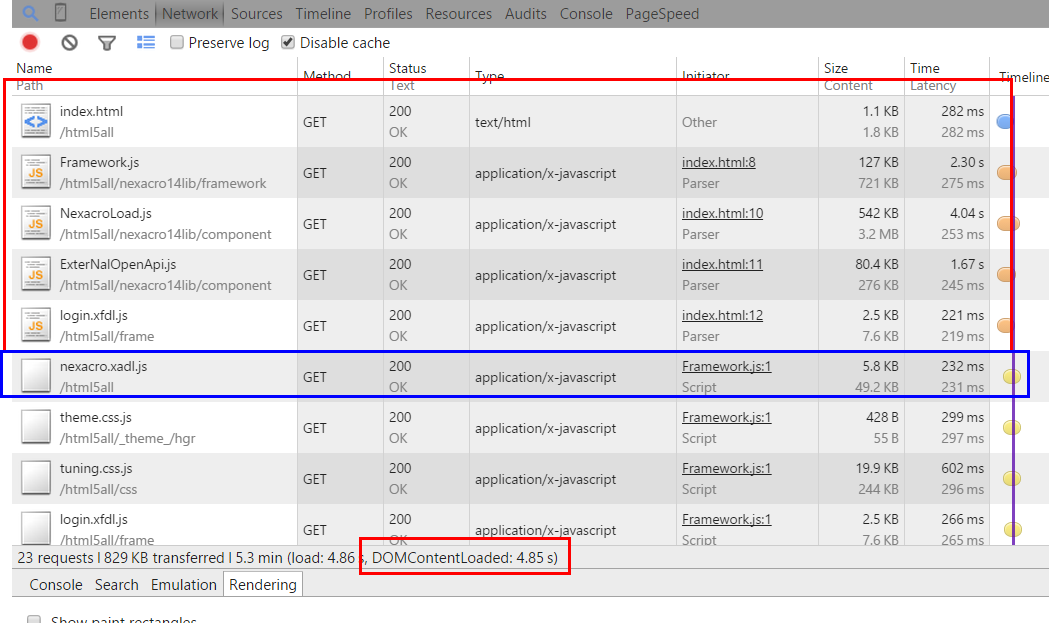
html은 Head 와 Body로 나뉘며 넥사크로에 Head에 걸려있는 넥사크로 unified framework 올라오고

body 에 adl이 올라오면서 화면을 그리게 되는 절차를 거칩니다.

### Timeline

빨간색으로 칠해져있는부분이 Header

파란색이 실제 body에 올라오는시간 즉 사용자게 체감하는시간은 DomContentLoaded 4.85초 + nexacro.xadl.js 올라온시간 232ms 가 사용자 체감시간이 됩니다. timeline에 빨간색과 노란색 경계선사이에 경계는 없어야합니다.



**Body**

**Header time**

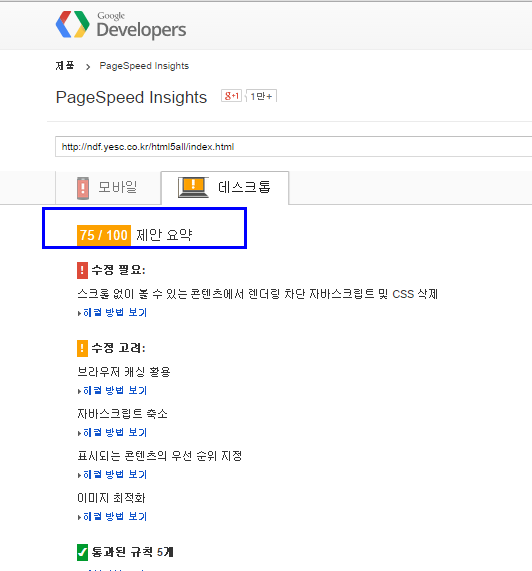
**Header**

## 성능측정

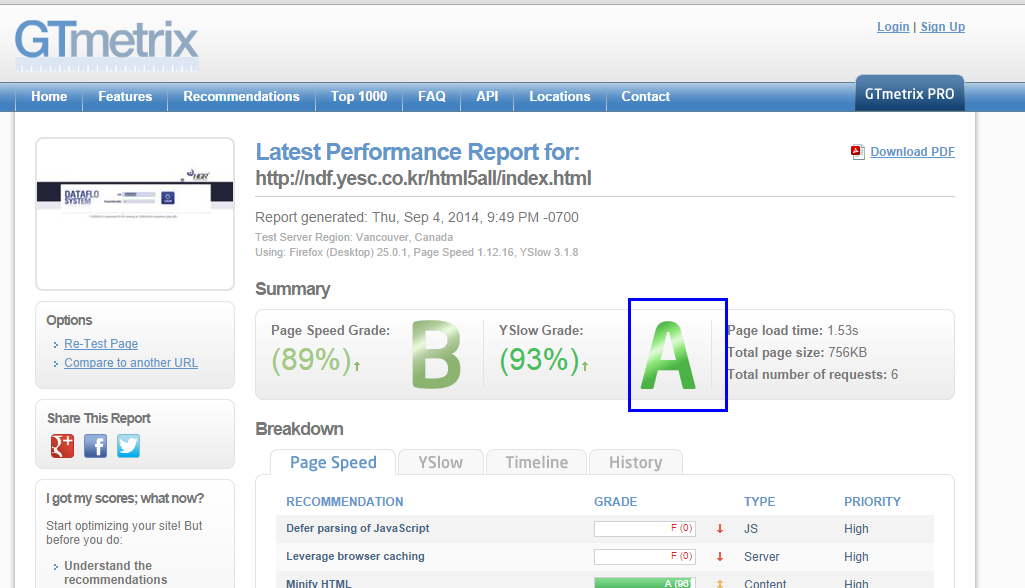
* PageSpeed Insights(URL: <https://developers.google.com/speed/pagespeed/insights/>)

Jascript 관련 페이징 테스트를 할수 있는 사이트입니다. 아래 참조사이트에 자체평가를 해주고 체적화를 할수 있는 가이드를 제공해 줍니다

PageSpeed insights



* GTmetrix(URL : <http://gtmetrix.com/reports/ndf.yesc.co.kr/6lIzzJIo>)



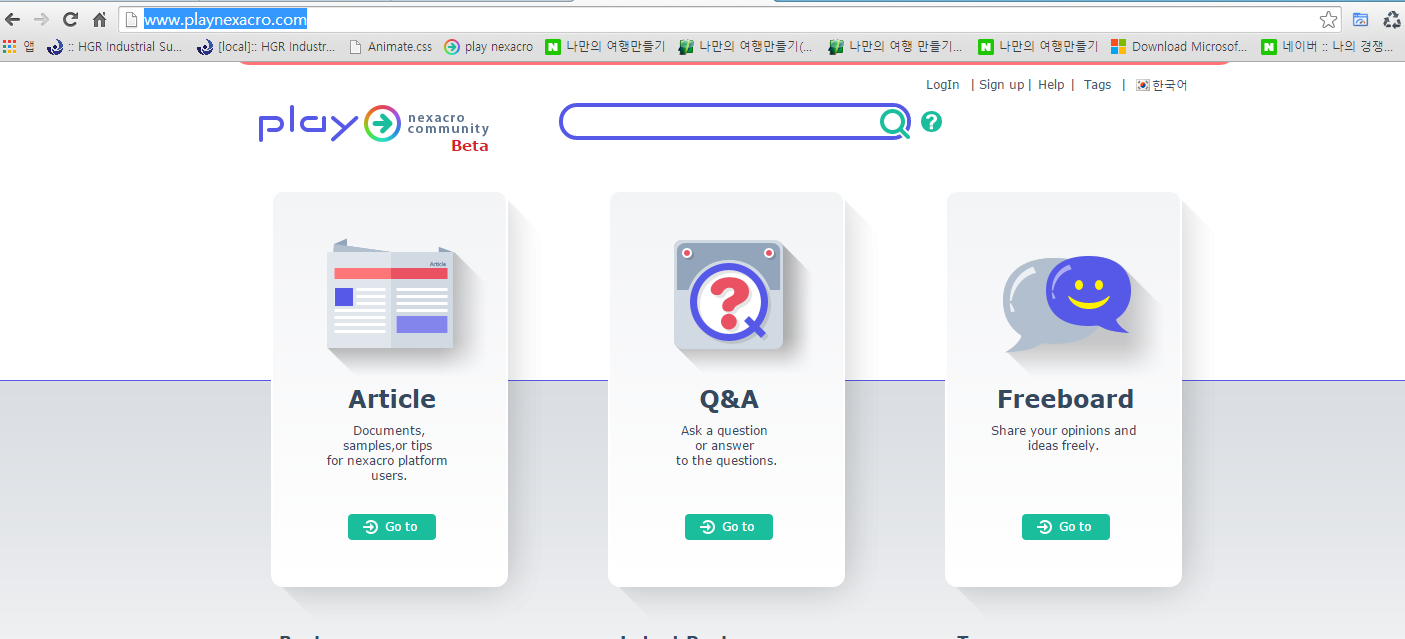
# playnexacro(플레이넥사크로)

url : [http://www.playnexacro.com/](http://www.playnexacro.com/%20)

-이 커뮤니티는 play nexacro라는 이름으로 일반적인 커뮤니티와 마찬가지로

nexacro platform에 관심있는 사람이면 누구나 자발적으로 참여할 수 있는 공간으로 활용하시면 되겠습니다.Nexacroplatform으로 구현에 필요한 여러 가지 유용한 정보들이 이곳에서 같이

논의되고, 공유되는 장소로 활용하시기 바랍니다.



# 웹디버거 사용

현재 공식사이트 위치 <http://www.fiddler2.com/fiddler2/>

Web Debugging 툴입니다. 웹에서 어떤 오류가 나왔을시.. 추적을 하거나 왜 그런일이 일어났는지. 도움받을수있습니다

