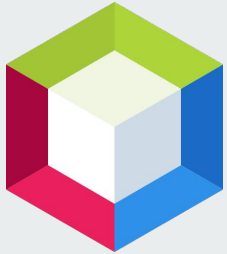
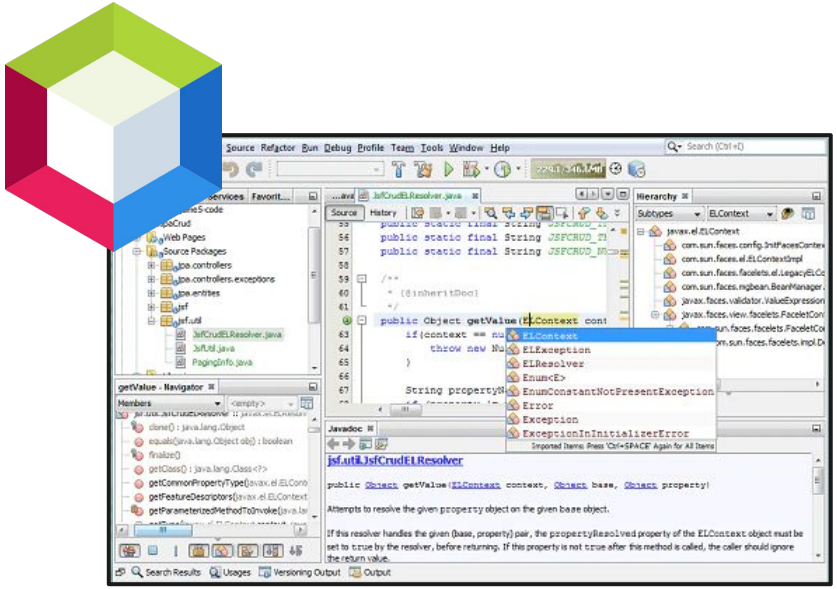
NetBeans IDE

Software Architecture Assignment



What is NetBeans IDE ?



Apache NetBeans

Is an

Integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called *modules*. NetBeans runs on Windows, macOS, Linux and Solaris. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML5 and JavaScript.

NetBeans - Oracle

What is NetBeans IDE Made of ?

The Apache NetBeans was made by **Java** and source code was donated by **Oracle** to the **Apache Software Foundation** in 2016.

NetBeans IDE Architecture

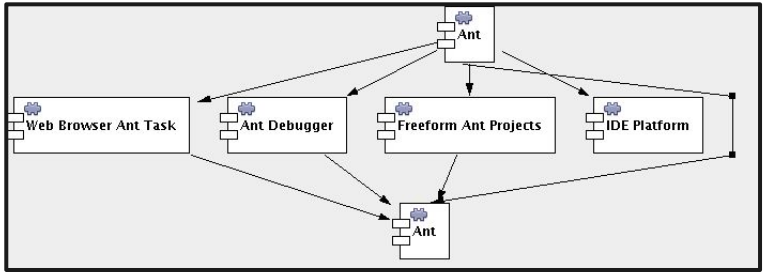
The big picture, a.k.a its architecture. The core of Apache NetBeans is its platform, or *Rich Client Platform* (RCP). The NetBeans IDE is built on top of the NetBeans RCP. In other words, one needs to understands the NetBeans RCP and its components in order to understand NetBeans.

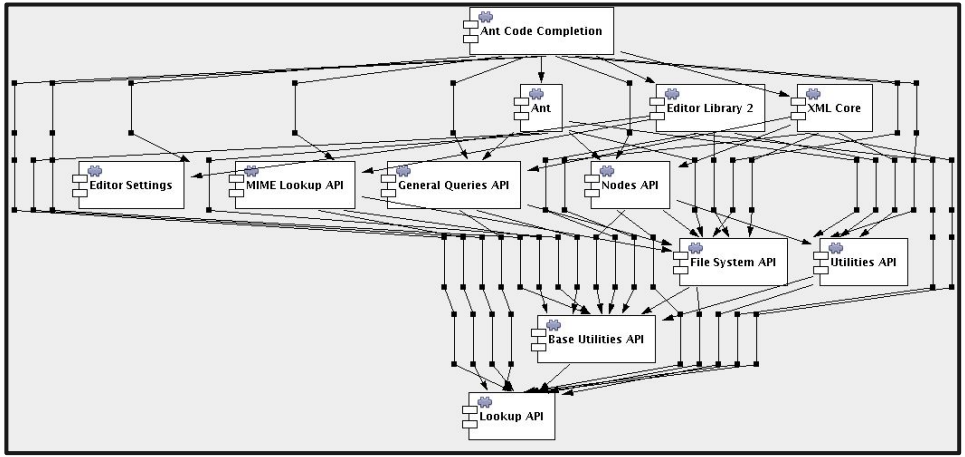
The NetBeans Platform consists of a number of features:

1. The module system
2. A Service infrastructure
3. A File System
4. A Window System
5. A Standardized UI Toolkit
6. A Generic Presentation Layer based on the Node API
7. Advanced Swing Components
8. JavaHelp Integration
9. A Lifecycle Management

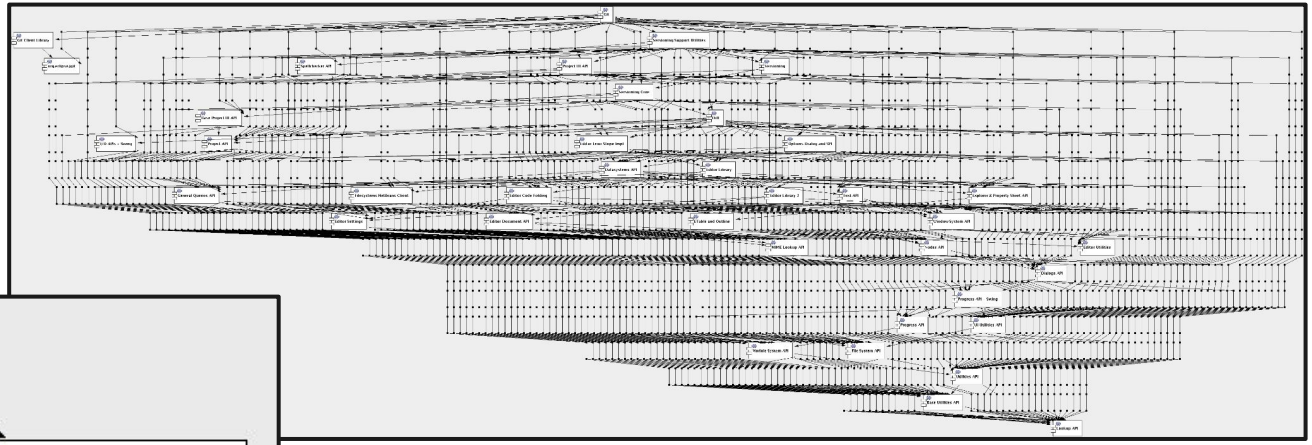
Netbeans IDE Architecture

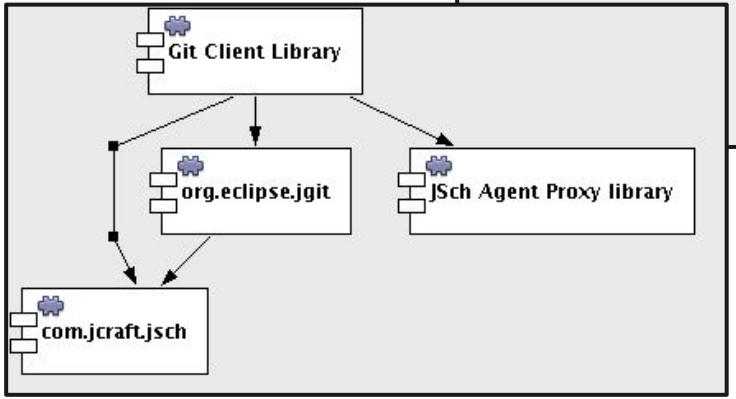
Below are the group of categories that contain more than 100 modules in the NetBeans platform.





And many more...





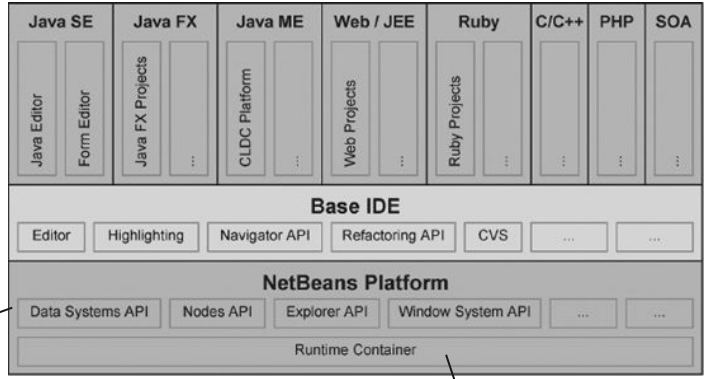
NetBeans

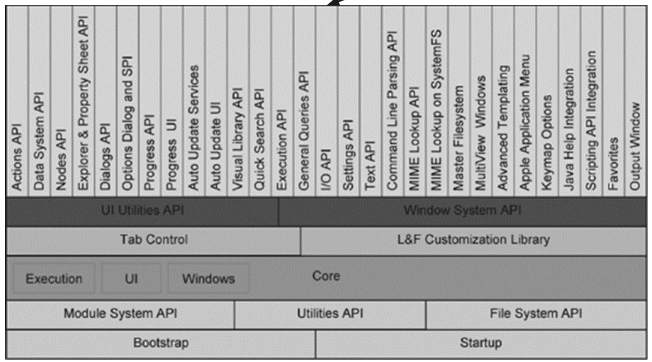
Definition of Netbeans Architecture

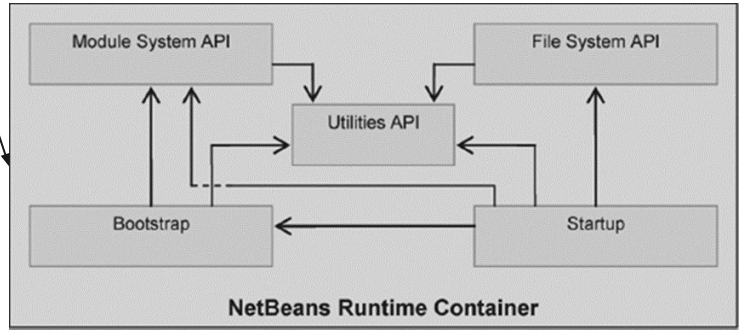
NetBeans architecture has concepts that are opposed to a monolithic architecture, in which every class makes use of code from any other class. The architecture is more flexible and simpler to maintain and support the development and conceptualization of flexible and **modular** applications.

NetBeans platform, as well as the application build on top of it, is divided into modules, of which each module is a **building block** making up a **modular architecture**. The modules are independent and have well-defined interfaces that are used by other parts of the same application. These are dynamically and automatically be loaded by the **core of the NetBeans runtime container,** after which it is responsible for running the application.

Definition of Netbeans Architecture (cont.)

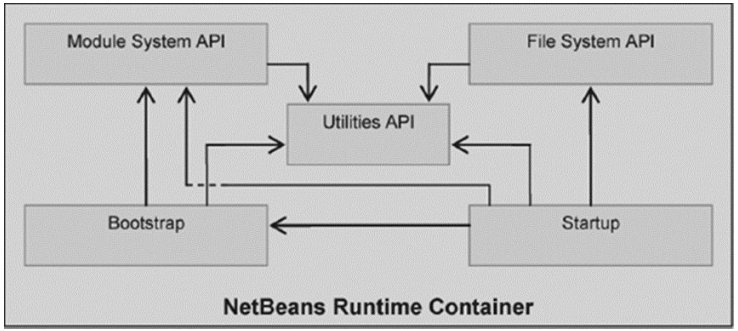


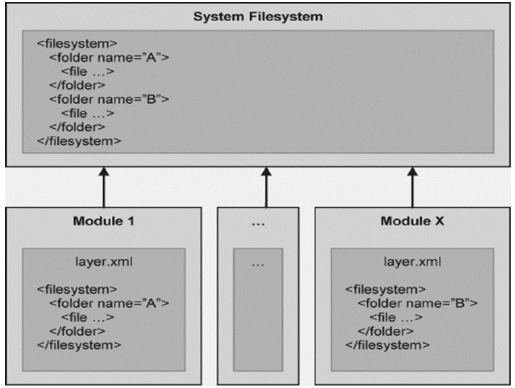


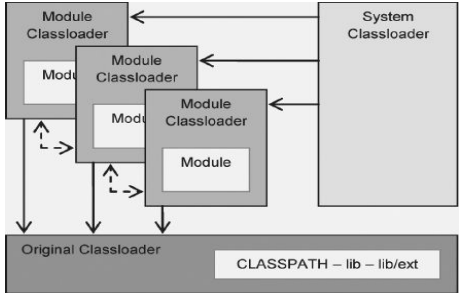


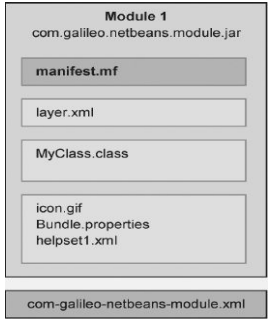
Guide to NetBeans™ Platform 7

Definition of Netbeans Architecture (cont.)



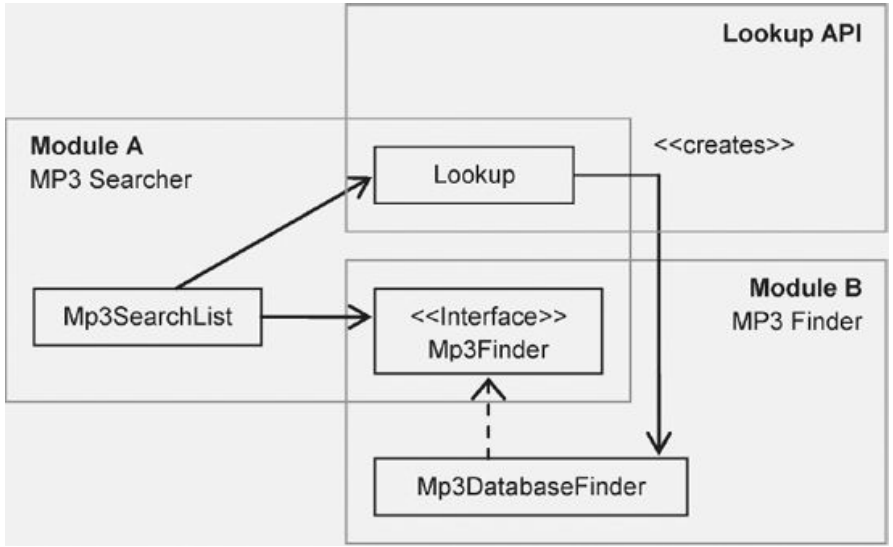


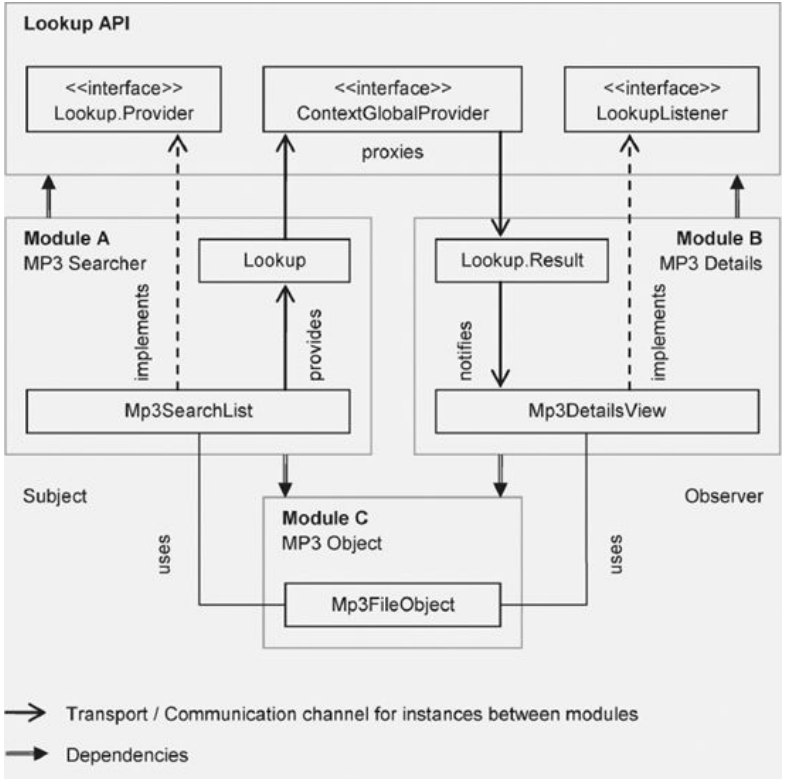




Guide to NetBeans™ Platform 7

Definition of Netbeans Architecture (cont.)

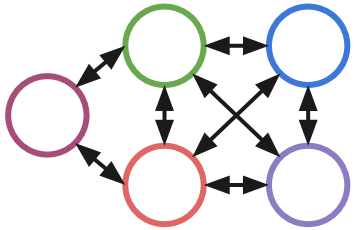




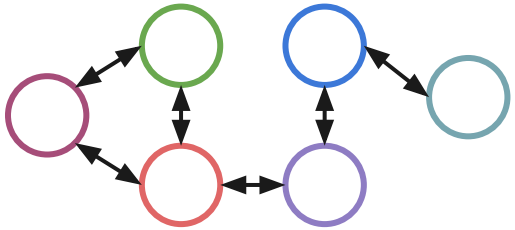
Advantages of the Architecture

NetBeans IDE is a very good example of a modular rich client application. The functionality and characteristics of an IDE, such as java language support, is created in the form of modules on top of the NetBeans Platform. That brings with it the great advantages that the application **can be extended** by additional modules and that it can be **adapted to specific user needs**, allowing particular modules to be deactivated or uninstalled.

NetBeans Platform enable modules to be **extendable** by other modules, and on the other hand enables them to communicate with each other being dependent on each other. In other words, NetBeans Platform **supports a loose coupling** of modules within an application.



Tightly Coupling



Loose Coupling

NetBeans Architecture Analysis

Quality Attribute

NetBeans IDE is a very good example of a **Modularity** application, NetBeans provides the ability to modify and manage which modules to includes and exclude unnecessary ones, that support **Extensibility**. With that to happen NetBeans support **Loose coupling** with every module in NetBeans is independent and exposed its well-defined interfaces to other modules, that support **Modifiability**. Additionally, some of these modules are completely reusable, which supports **Reusability**.

Architectural Pattern

According to the Quality Attributes that were mentioned in the Quality Attributes section. The particular Architectural Pattern that represents and reflect all that has been mentioned is the one call **Model-View-Controller (MVC)** in which

Model is represented by all the modules that act as a functionality of the NetBeans.

View is represented by all the modules that the user can interact with of the NetBeans.

Controller is represented by the core of the NetBeans runtime container that loads relevant modules to perform any runtime application.

NetBeans Architecture Analysis (cont.)

Architectural Disadvantage and Strategy

The Great weakness of MVC is complexity. The MVC architecture is sometime hard to understand and take time to read the document and strict rules.

The isolated development process by UI authors, business logic authors and controller authors may lead to delay in their respective modules development.

To counter such weakness, We rise some strategy to help reduces the weakness of the MVC

- Take time write some documentations, always comment on the code sections and provide some example of how to use any module.
- Well design and implement is needed.
- Use version controller

Strategy

แยก Algorithm ที่คล้ายกันไปแต่ละ Class เพื่อให้ง่ายต่อการ Implement ที่แตกต่างกันเล็กน้อย

👍 ข้อดี

การนำ Strategy Pattern มาใช้งานนั้น

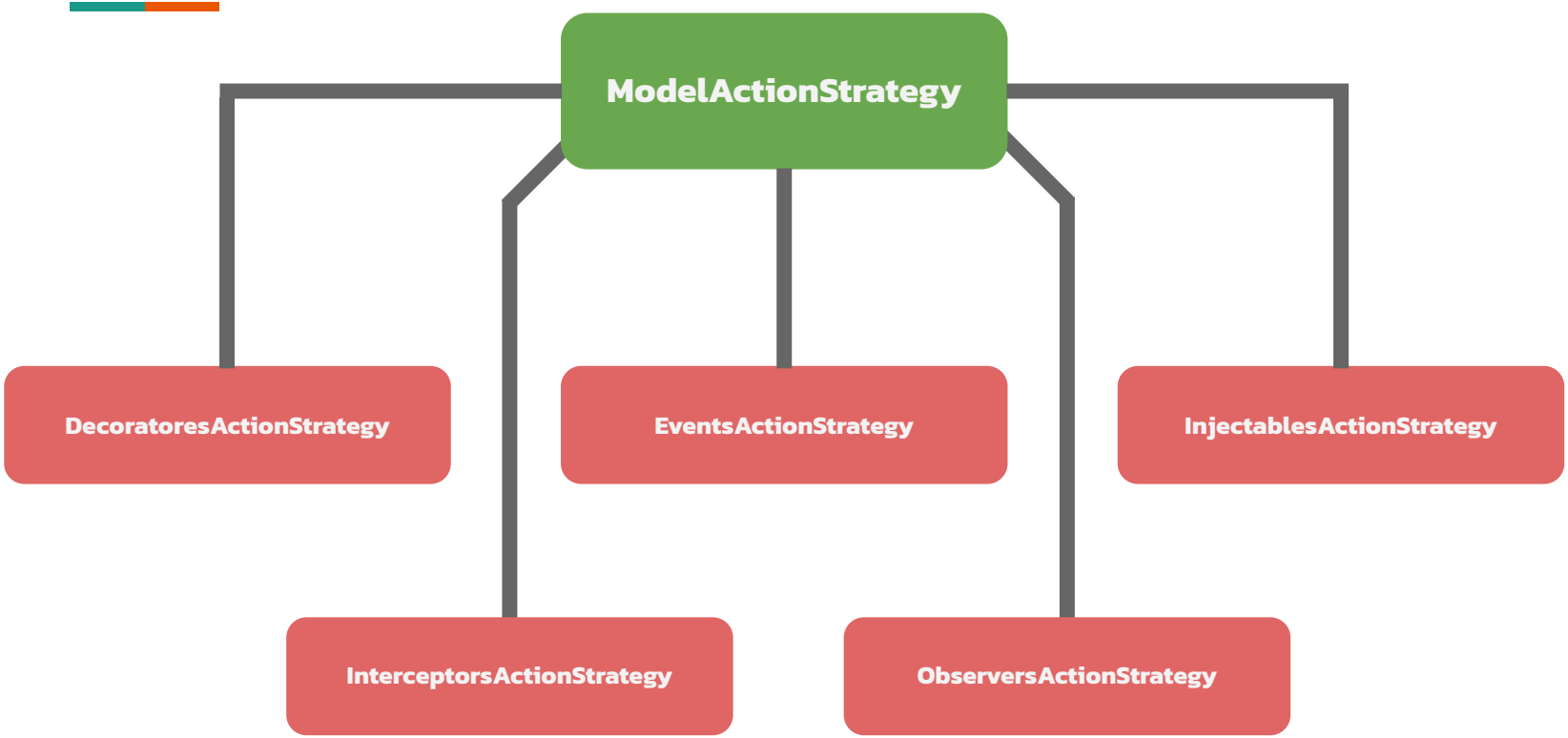
- ช่วยให้ง่ายต่อการเพิ่ม Algorithm ใหม่ ๆ ที่มีการเปลี่ยนการทำงานเล็กน้อย

  หรือก็คือมีความยืดหยุ่นมากยิ่งขึ้น
- แก้ปัญหาการเกิด Open/Closed Principle

Link to diagram

https://drive.google.com/file/d/1ngfql8js6nTP7EO4Bxsk2fUM39nJDEr/view?usp=sharing

```
32  public interface ModelActionStrategy {
33
34      public enum InspectActionId {
35          OBSERVERS_CONTEXT,
36          METHOD_CONTEXT,
37          INJECTABLES_CONTEXT,
38          CLASS_CONTEXT;
39      }
40
41      boolean isApplicable( InspectActionId id );
42
43      boolean isApplicable( WebBeansModel model , Object[] context );
44
45      void invokeModelAction( WebBeansModel model,
46              MetadataModel<WebBeansModel> metaModel, Object[] subject,
47              JTextComponent component, FileObject fileObject  );
48  }
```


ModelActionStrategy
DecoratoresActionStrategy
EventsActionStrategy
InjectablesActionStrategy
InterceptorsActionStrategy
ObserversActionStrategy

```java
43  public final class DecoratoresActionStrategy implements ModelActionStrategy {
44
45      /* (non-Javadoc)
46       * @see org.netbeans.modules.web.beans.navigation.actions.ModelActionStrategy#isApplicable(org.netbeans.modules.web.beans.
47       */
48      @Override
49      public boolean isApplicable( InspectActionId id ) {
50          return id == InspectActionId.CLASS_CONTEXT;
51      }
52
53      /* (non-Javadoc)
54       * @see org.netbeans.modules.web.beans.navigation.actions.ModelActionStrategy#isApplicable(org.netbeans.modules.web.beans.
55       */
56      @Override
57      public boolean isApplicable( WebBeansModel model, Object[] context ) {
58          final Object handle = context[0];
59          if ( handle == null ){
60              return false;
61          }
62          Element element = ((ElementHandle<?>)handle).resolve(
63                  model.getCompilationController());
64          if ( element == null ){
65              return false;
66          }
67          List<AnnotationMirror> qualifiers = model.getQualifiers(element,  true);
68          // if class has qualifiers then Class context is considered as decorator context
69          if ( qualifiers.size() >0 ){
70              return true;
71          }
72          /*
73           *  If it doesn't have explicit qualifiers then it could have implicit @Default
74           *  qualifier . In the latter case check Interceptor Bindings presence
75           */
76          return model.getInterceptorBindings(element).isEmpty();
77      }
78
79      /* (non-Javadoc)
80       * @see org.netbeans.modules.web.beans.navigation.actions.ModelActionStrategy#invokeModelAction(org.netbeans.modules.web.b
81       */
82      @Override
83      public void invokeModelAction( WebBeansModel model,
84              final MetadataModel<WebBeansModel> metaModel, final Object[] subject,
85              JTextComponent component, FileObject fileObject )
86      {
87          final Object handle = subject[0];
88          Element element = ((ElementHandle<?>)handle).resolve(
```

```java
42  public final class InjectablesActionStrategy implements ModelActionStrategy {
43
44      /* (non-Javadoc)
45       * @see org.netbeans.modules.web.beans.navigation.actions.ModelActionStrategy#isApplicable(org.netbeans.modules.web.beans
46       */
47      @Override
48      public boolean isApplicable( InspectActionId id ) {
49          return id == InspectActionId.INJECTABLES_CONTEXT ;
50      }
51
52      @Override
53      public boolean isApplicable( WebBeansModel model, Object context[] ) {
54          final VariableElement var = WebBeansActionHelper.findVariable(model, context);
55          if (var == null) {
56              return false;
57          }
58          try {
59              if ( model.isEventInjectionPoint(var)){
60                  return false;
61              }
62              if (!model.isInjectionPoint(var)) {
63                  StatusDisplayer.getDefault().setStatusText(
64                          NbBundle.getMessage(GoToInjectableAtCaretAction.class,
65                                  "LBL_NotInjectionPoint"), // NOI18N
66                          StatusDisplayer.IMPORTANCE_ERROR_HIGHLIGHT);
67                  return false;
68              }
69          }
70          catch (InjectionPointDefinitionError e) {
71              StatusDisplayer.getDefault().setStatusText(e.getMessage(),
72                      StatusDisplayer.IMPORTANCE_ERROR_HIGHLIGHT);
73          }
74          return true;
75      }
76
77      /* (non-Javadoc)
78       * @see org.netbeans.modules.web.beans.navigation.actions.ModelActionStrategy#invokeModelAction(org.netbeans.modules.web
79       */
80      @Override
81      public void invokeModelAction( final WebBeansModel model,
82              final MetadataModel<WebBeansModel> metaModel, final Object[] subject,
83              JTextComponent component, FileObject fileObject )
84      {
85          final VariableElement var = WebBeansActionHelper.findVariable(model,
86                  subject);
87          DependencyInjectionResult result = var== null? null: model.lookupInjectables(var, null, new AtomicBoolean(false));
```

```java
50  public class InspectCDIAtCaretAction extends AbstractWebBeansAction {
51
52      private static final long serialVersionUID = -4505119467924502377L;
53
54      private static final String INSPECT_CDI_AT_CARET =
55          "inspect-cdi-at-caret";                     // NOI18N
56
57      private static final String INSPECT_CDI_AT_CARET_POPUP =
58          "inspect-cdi-at-caret-popup";               // NOI18N
59
60      public InspectCDIAtCaretAction( ) {
61          super(NbBundle.getMessage(InspectCDIAtCaretAction.class,
62                  INSPECT_CDI_AT_CARET));
63          myStrategies = new ArrayList<ModelActionStrategy>( 4 );
64          /*
65           *  The order is important !
66           *  EventsActionStartegy should be after InjectablesActionStrategy
67           *  because it cares about several action ids.
68           */
69          myStrategies.add( new ObserversActionStrategy());
70          myStrategies.add( new InjectablesActionStrategy());
71          myStrategies.add( new DecoratoresActionStrategy());
72          myStrategies.add( new InterceptorsActionStrategy());
73          myStrategies.add( new EventsActionStartegy());
74      }
```

Builder

- แยกการสร้าง object ที่ต้องอาศัยขั้นตอนการสร้างที่ซับซ้อนออก และช่วยให้เราสามารถสร้าง object ประเภทอื่นๆที่มีขั้นตอนการสร้างแบบเดียวกันได้
- ช่วยให้ โค้ดขั้นตอนการสร้างที่เหมือนๆกันรวมอยู่ที่เดียวกัน แต่สามารถสร้าง object ที่แตกต่างกันได้

👍 ข้อดี

การนำ **Builder Pattern** มาใช้งานนั้นจะช่วยลดการผูกกันของโค้ดลง ทำให้เราสามารถเปลี่ยนแปลง แก้ไข รองรับสิ่งต่างๆได้มากขึ้น แถมยังช่วยลดโค้ดการสร้าง object ที่มีขั้นตอนในการสร้างเหมือนๆกันได้อีกด้วย

Link to diagram

https://drive.google.com/file/d/1Tut-rqz1FlbOnBv1tGGkB_hqhw63sqtv/view?usp=sharing

```java
    public static final class JobData {

        private String jobName;
        private String jobUrl;
        private boolean secured;
        private HudsonJob.Color color;
        private String displayName;
        private boolean buildable;
        private boolean inQueue;
        private int lastBuild;
        private int lastFailedBuild;
        private int lastStableBuild;
        private int lastSuccessfulBuild;
        private int lastCompletedBuild;
        private List<ModuleData> modules = new LinkedList<ModuleData>();
        private List<String> views = new LinkedList<String>();

        public String getJobName() {
            return jobName;
        }

        public void setJobName(String jobName) {
            this.jobName = jobName;
        }

        public String getJobUrl() {
            return jobUrl;
        }

        public void setJobUrl(String jobUrl) {
            this.jobUrl = jobUrl;
        }
```

ide/hudson/test/unit/src/org/netbeans/modules/hudson/impl/HudsonInstanceImplTest.java

```java
80          public BuilderConnector.InstanceData getInstanceData(
81                  boolean authentication) {
82              JobData jd = new JobData();
83              jd.setJobName("test");
84              jd.setJobUrl("http://x230406/job/test/");
```

Java Showing the top match Last indexed on Mar 24

```java
80          public BuilderConnector.InstanceData getInstanceData(
81                  boolean authentication) {
82              JobData jd = new JobData();
83              jd.setJobName("test");
84              jd.setJobUrl("http://x230406/job/test/");
85              ViewData wd = new ViewData("Main", "http://x230406/main/",
86                      true);
87              return new InstanceData(Collections.singleton(jd),
88                      Collections.singleton(wd),
89                      Collections.<FolderData>emptySet());
90          }
```

Singleton

จำกัดให้มีการสร้าง Object เพียงตัวเดียวไม่ถูกสร้างซ้ำ เพื่อไม่ให้เกิดความซ้ำซ้อน

👍 ข้อดี

- ช่วยให้เราสามารถควบคุมการสร้าง object ได้
- มีช่องทางให้เข้าถึงแบบ Global
- ซ่อนความวุ่นวายในการสร้าง object
- ถ้าการสร้าง object มีการเปลี่ยนแปลง ก็สามารถแก้ได้จากจุดเดียว

Link to diagram

https://drive.google.com/file/d/1c5u8TY4gHpqWxBCSQkK96fXsRgRNXoSm/view?usp=sharing

```java
    private static ServerRegistry registry;
```

```java
    private ServerRegistry(String path, boolean cloud) {
        this.path = path;
        this.cloud = cloud;
        lookup = Lookups.forPath(path);
        result = lookup.lookupResult(ServerInstanceProvider.class);
    }
```

```java
    public static synchronized ServerRegistry getInstance() {
        if (registry == null) {
            registry = new ServerRegistry(SERVERS_PATH, false);
            registry.result.allItems();
            registry.result.addLookupListener(l = new ProviderLookupListener(registry.changeSupport));
        }
        return registry;
    }
```

enterprise/j2eeserver/test/unit/src/org/netbeans/modules/j2ee/deployment/impl/ServerRegistryTest.java

```java
47      public void testPluginLayerFile() {
48          ServerRegistry registry = ServerRegistry.getInstance();
49          System.out.println ("registry:" + registry);
...
84      public void testDeploymentFileNames() {
85          ServerRegistry registry = ServerRegistry.getInstance();
86          Server testPlugin = registry.getServer("Test");
```

Java Showing the top four matches Last indexed on Mar 24

Observer

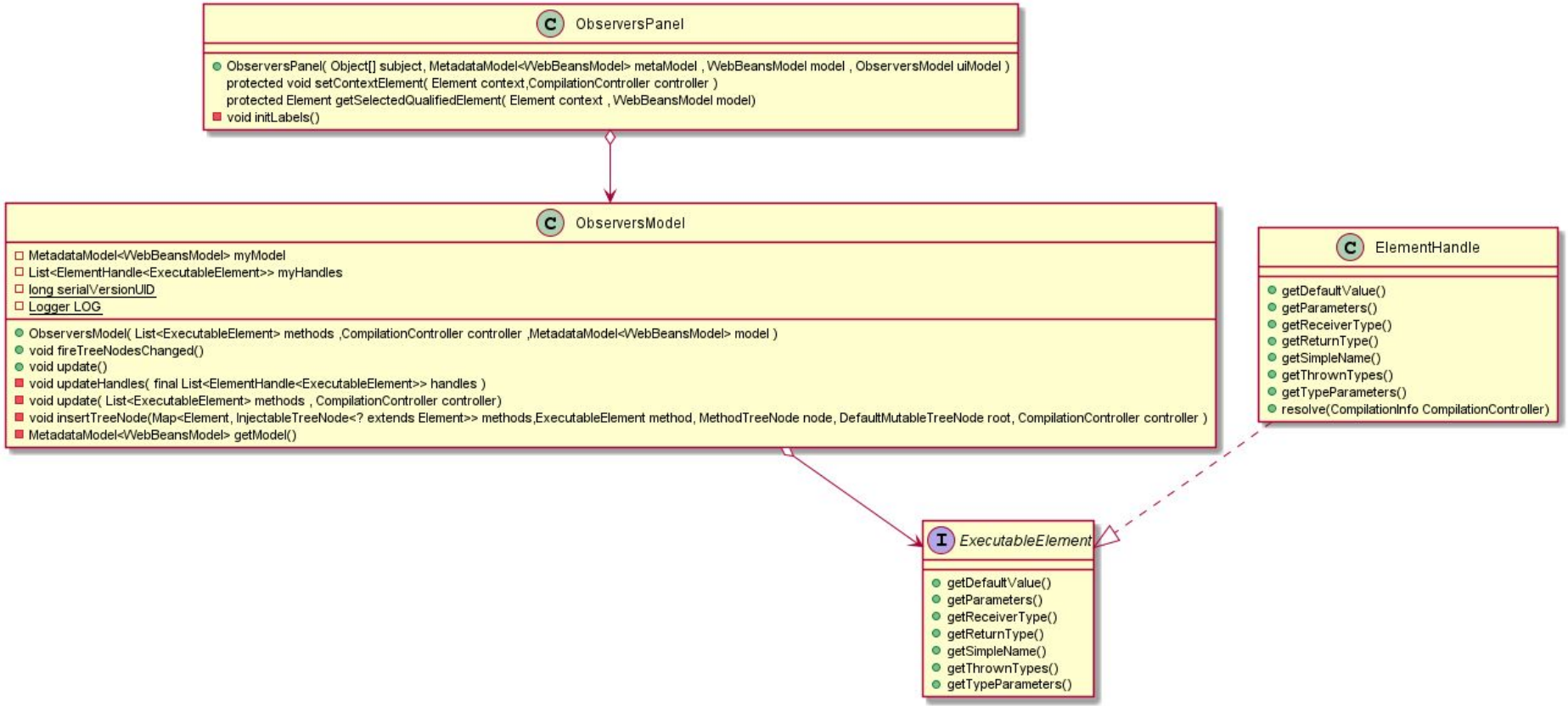
ใช้ Subscription mechanism เพื่อใช้ในการส่งข้อมูลให้กับหลาย Object ที่ Subscribe ไว้

👍 ข้อดี

- ส่งการแจ้งเตือนให้กับคนที่สนใจจะรับข่าวสารนั้นจริงๆ
- แก้ปัญหาการเกิด Open/Closed Principle โดยสามารถเพิ่ม Subscriber Class ขึ้นมาใหม่ได้โดยไม่จำเป็นต้องเปลี่ยนอะไรใน Publisher Class

Link to diagram

https://drive.google.com/file/d/18zq0twvpVpNuB97AKkghNM1RUfkxJ9Tu/view?usp=sharing


ObserversPanel
ObserversPanel(Object[] subject, MetadataModel<WebBeansModel> metaModel , WebBeansModel model , ObserversModel uiModel)
protected void setContextElement(Element context,CompilationController controller)
protected Element getSelectedQualifiedElement(Element context , WebBeansModel model)
void initLabels()
ObserversModel
MetadataModel<WebBeansModel> myModel
List<ElementHandle<ExecutableElement>> myHandles
long serialVersionUID
Logger LOG
ObserversModel(List<ExecutableElement> methods ,CompilationController controller ,MetadataModel<WebBeansModel> model)
void fireTreeNodesChanged()
void update()
void updateHandles(final List<ElementHandle<ExecutableElement>> handles)
void update(List<ExecutableElement> methods , CompilationController controller)
void insertTreeNode(Map<Element, InjectableTreeNode<? extends Element>> methods,ExecutableElement method, MethodTreeNode node, DefaultMutableTreeNode root, CompilationController controller)
MetadataModel<WebBeansModel> getModel()
ElementHandle
getDefaultValue()
getParameters()
getReceiverType()
getReturnType()
getSimpleName()
getThrownTypes()
getTypeParameters()
resolve(CompilationInfo CompilationController)
ExecutableElement
getDefaultValue()
getParameters()
getReceiverType()
getReturnType()
getSimpleName()
getThrownTypes()
getTypeParameters()

```
59      public ObserversModel( List<ExecutableElement> methods ,
60              CompilationController controller ,MetadataModel<WebBeansModel> model )
61      {
62          super( null );
63          myModel = model;
64          myHandles = new ArrayList<ElementHandle<ExecutableElement>>( methods.size());
65          for (ExecutableElement method : methods) {
66              ElementHandle<ExecutableElement> handle = ElementHandle.create( method );
67              myHandles.add( handle );
68          }
69
70          update( methods , controller );
71      }
```

```
86      public void update() {
87          updateHandles( myHandles );
88      }
89
90      private void updateHandles( final List<ElementHandle<ExecutableElement>> handles ) {
91          try {
92              getModel().runReadAction(
93                      new MetadataModelAction<WebBeansModel, Void>() {
94
95                          public Void run( WebBeansModel model ) {
96                              List<ExecutableElement> list =
97                                  new ArrayList<ExecutableElement>(handles.size());
98                              for (ElementHandle<ExecutableElement> handle :
99                                  handles)
100                             {
101                                 ExecutableElement method = handle.resolve(
102                                         model.getCompilationController());
103                                 if ( method != null ){
104                                     list.add( method );
105                                 }
106                             }
107                             update( list , model.getCompilationController());
108                             return null;
109                         }
110                     });
111
112             return;
113         }
```

OAT

ภูบดินทร์ เทียนทอง
62010711

OAK

วงศ์วริศ พันธ์เจริญ
62010787

POR

วรนิพิฐ ชัยพินิจนรชาติ
62010794

DRAGON

วีรวิชญ์ ศรีสันติคณาพร
62010838

BAO

ศิวกร เพชรน้อย
62010882

ขอบคุณ