Eclipse 4 Goes Formal: API You Can Rely On

- Eric Moffatt
  - Platform UI and e4 comitter
  - Eclipse 4 Dev Lead
  - “Pixel Pusher” I focus on the UI bits; renderers, Min/Max...

What do we use API for anyway ?

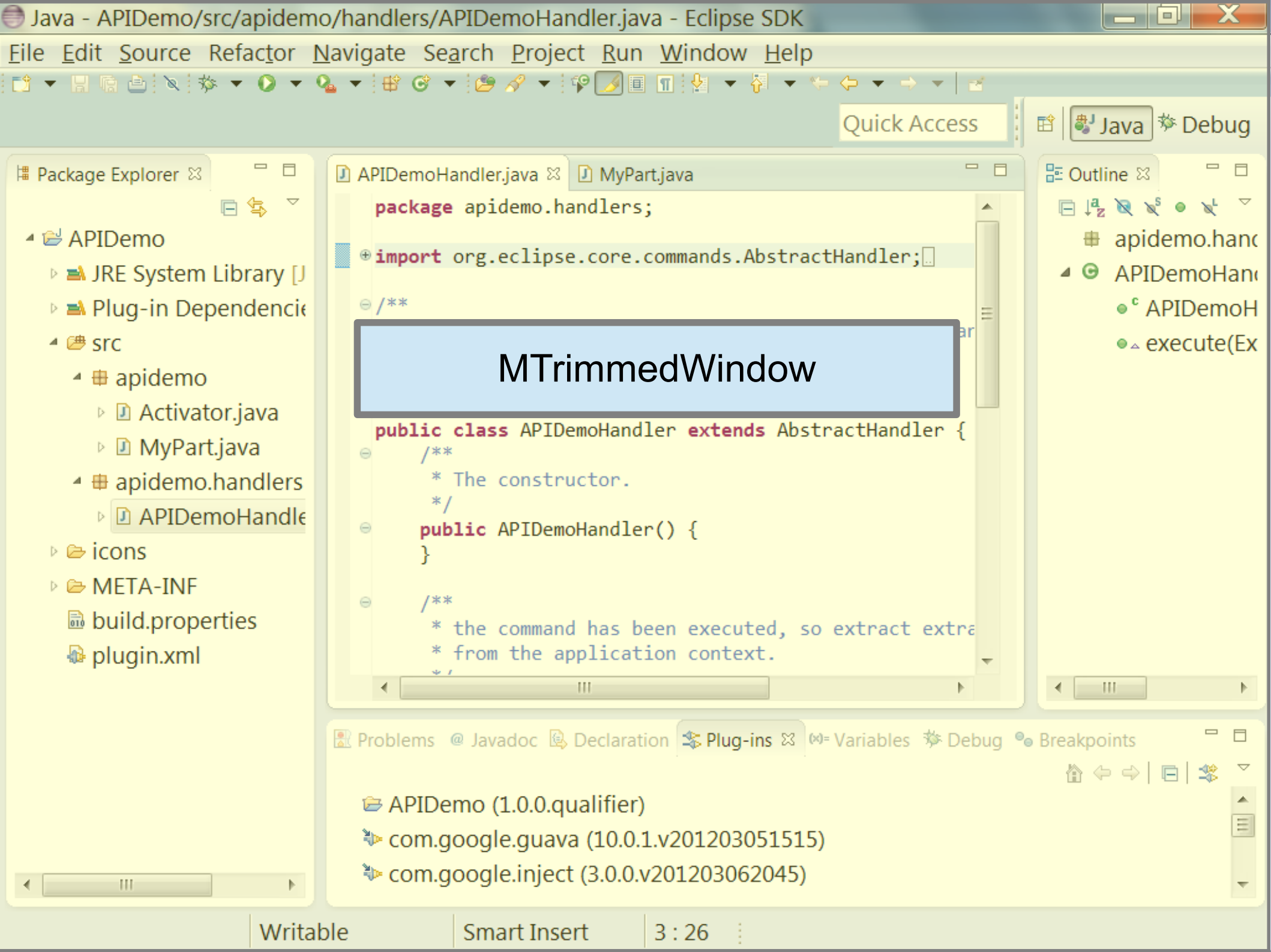
- Getting new stuff into the UI...
  - Views / Editors
  - Menus / Toolbars and their Items
  - Trim Controls
  - Perspectives
- … and getting it to work
  - Accessing services
  - Listening for changes in UI state

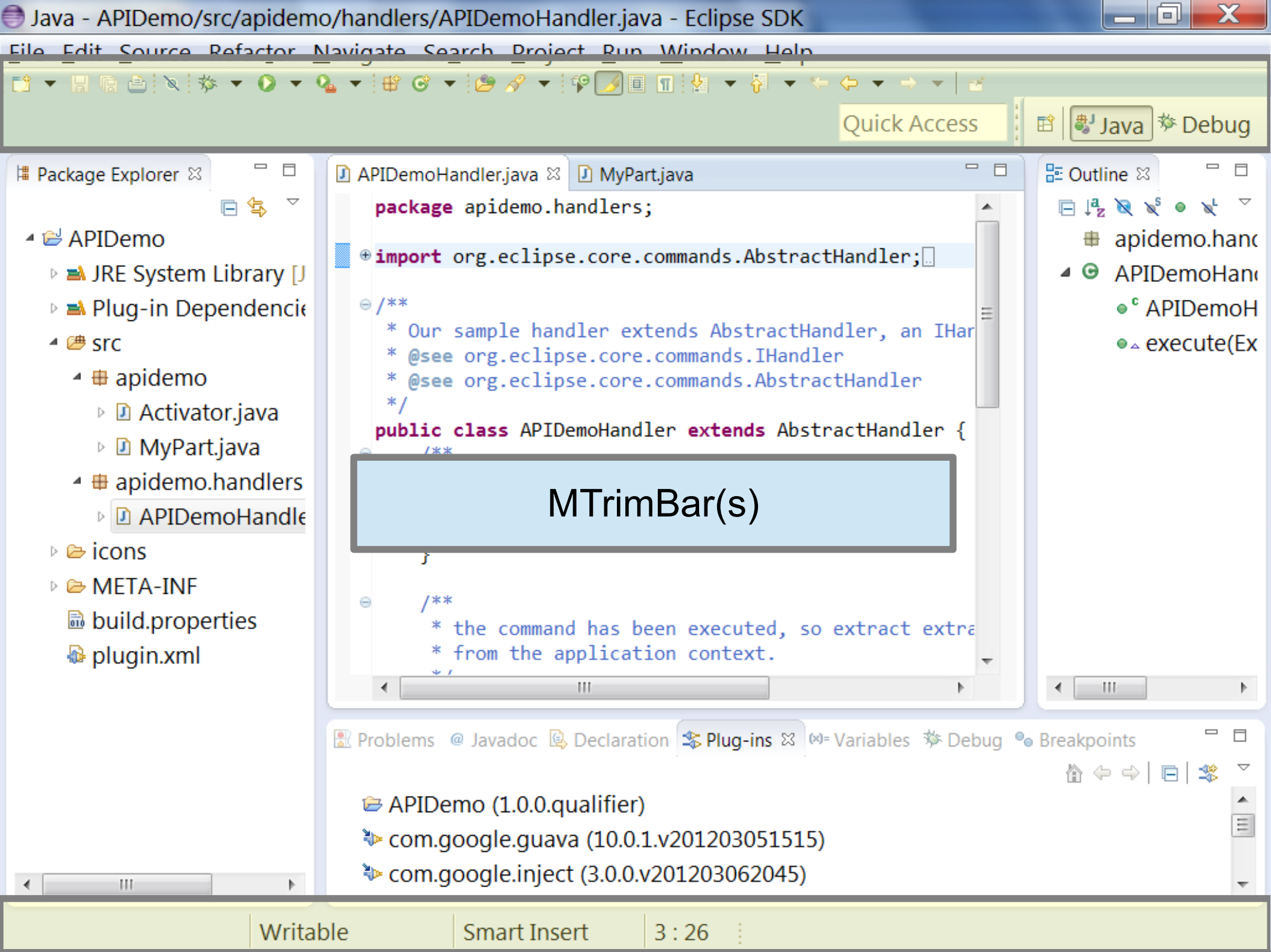
Eclipse 3 API (aka 'Old School')

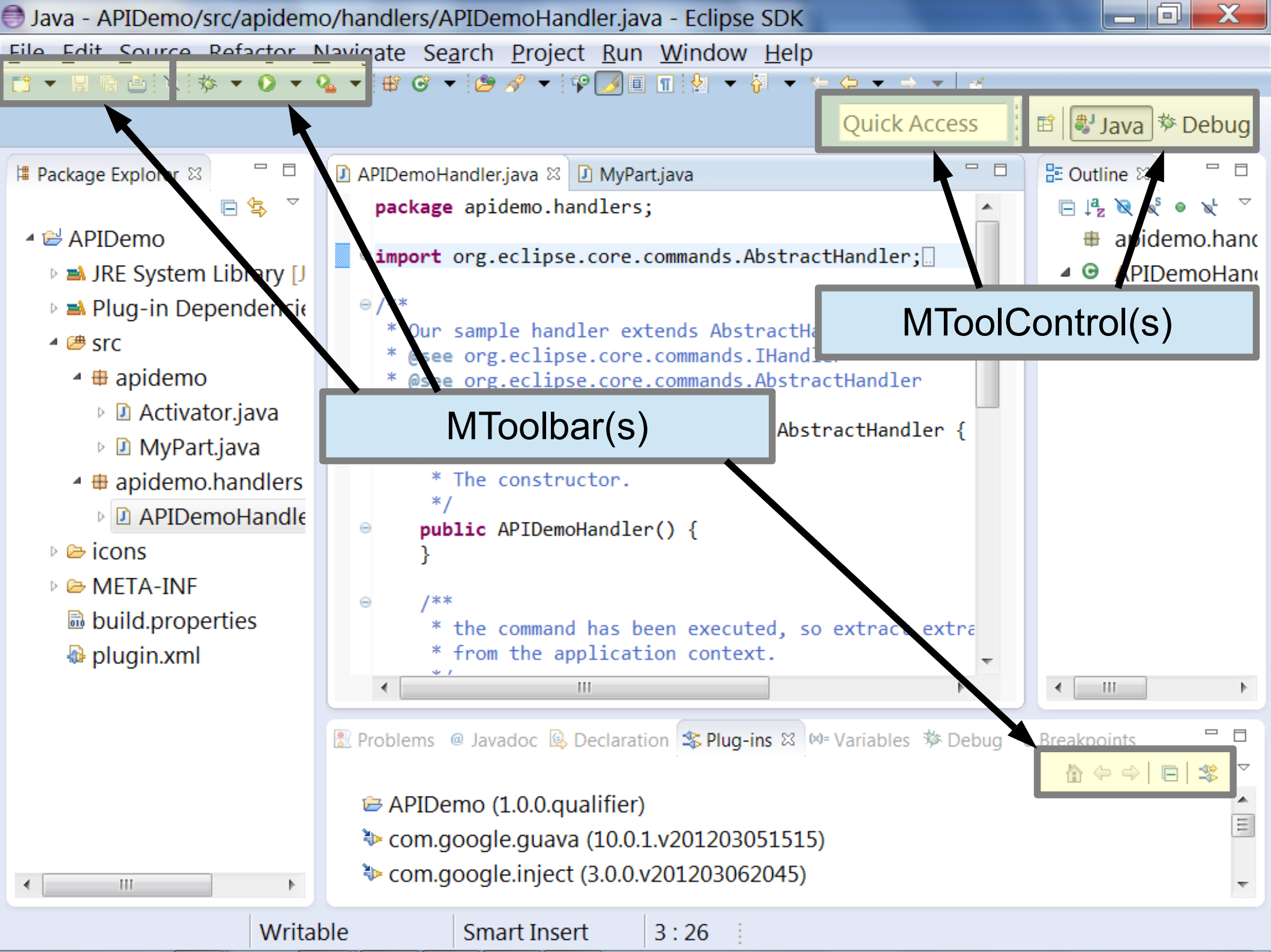
- Organically grown over many years
- Single points of access means inheriting lots of dependencies you really didn't need
  - IWorkbench[Window | Page] are used as access points meaning that while you only want to consume one service you end up dragging in *every* type referenced.
- Uses singletons (i.e. PlatformUI)
  - Bad for multi-instance apps...

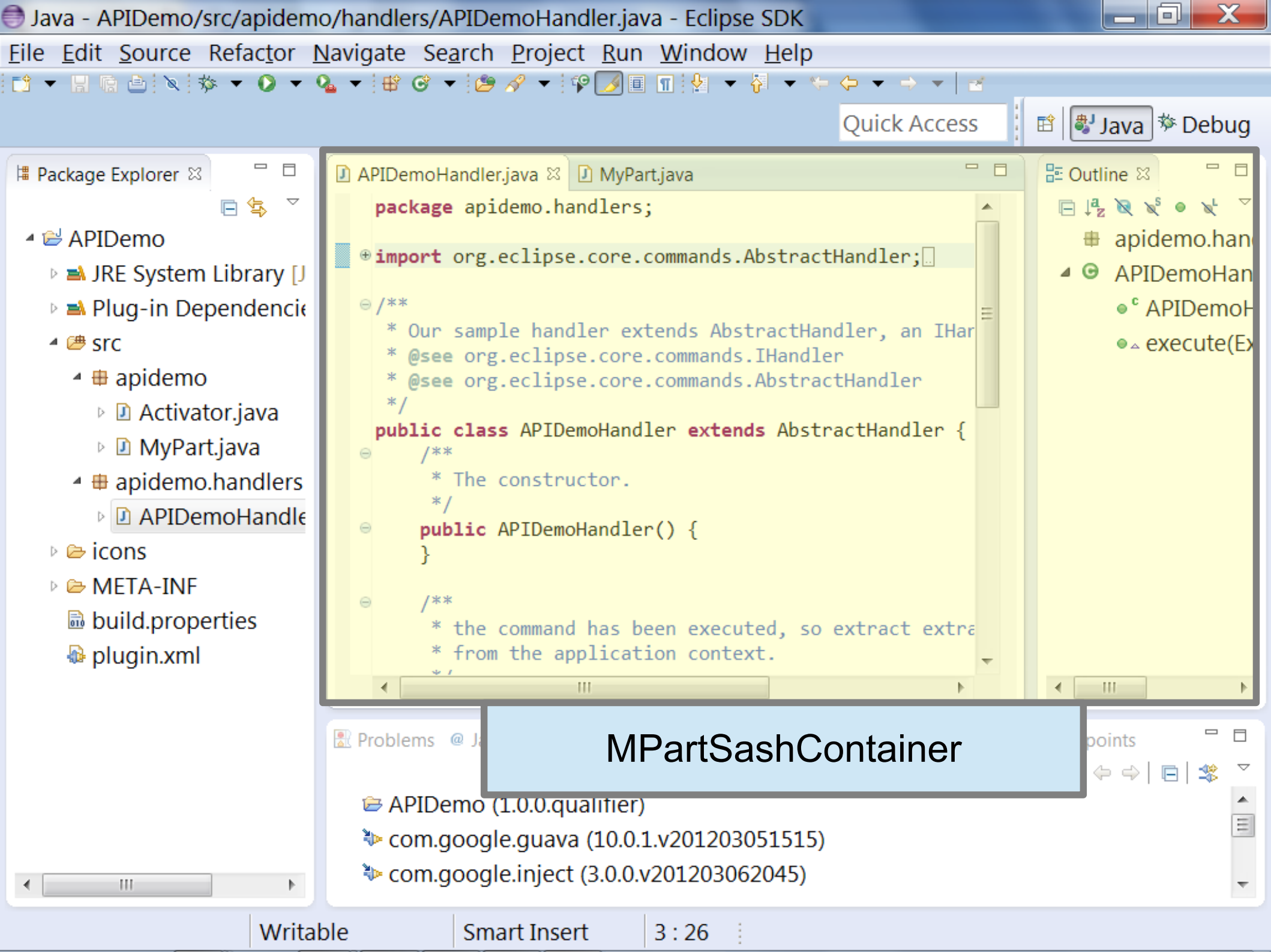
Eclipse 4 API

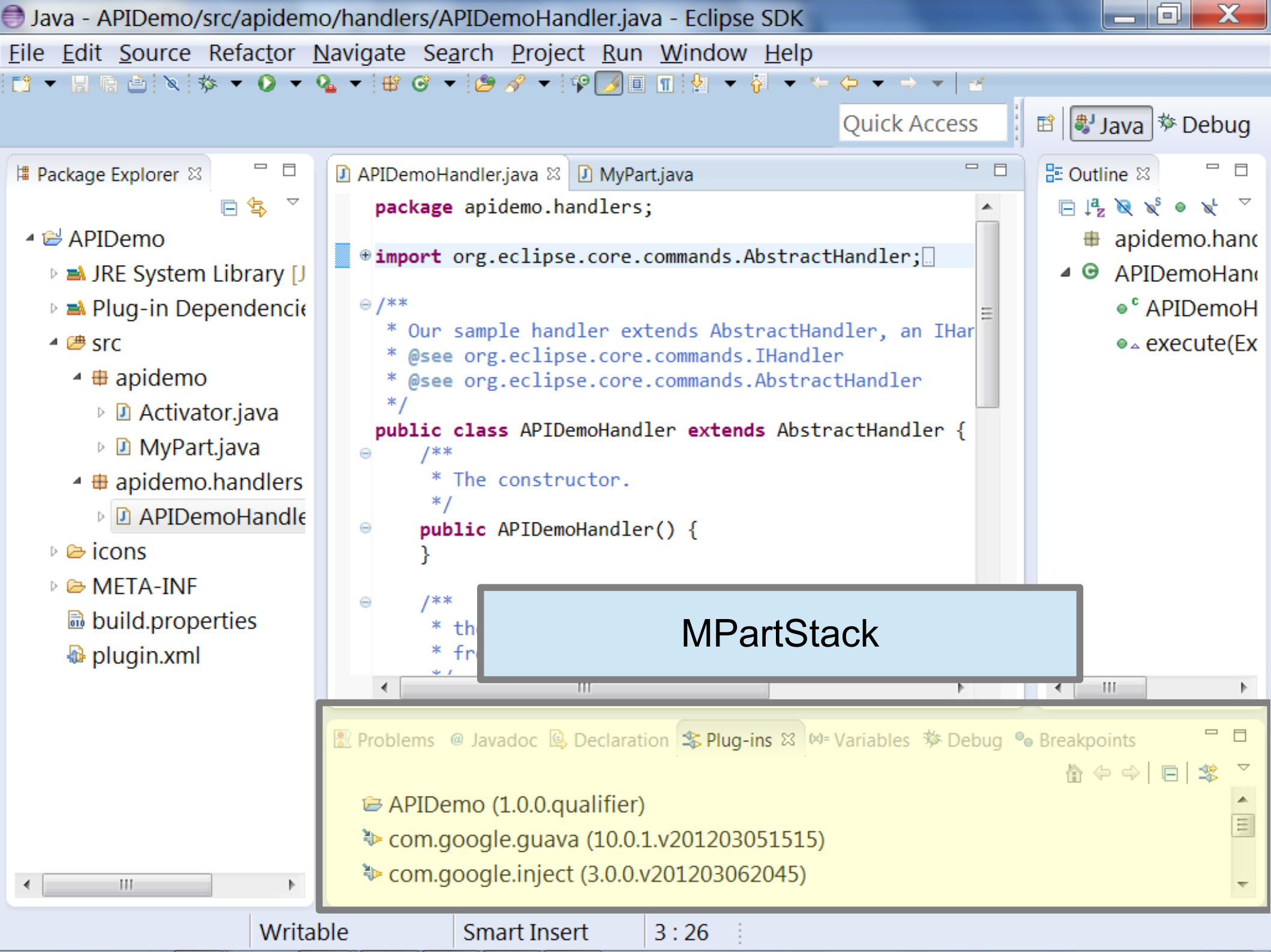
- Two general mechanisms cover everything
  - UI Model (How you talk to us)l
    - Defines what you see on the screen
    - Sends events for *all* model changes
  - Dependency Injection (How we talk to you)
    - You just describe what you need
    - How you access services
    - Dynamic re-injection (replaces most listeners)
    - You describe what you require using annotations...but only what you need

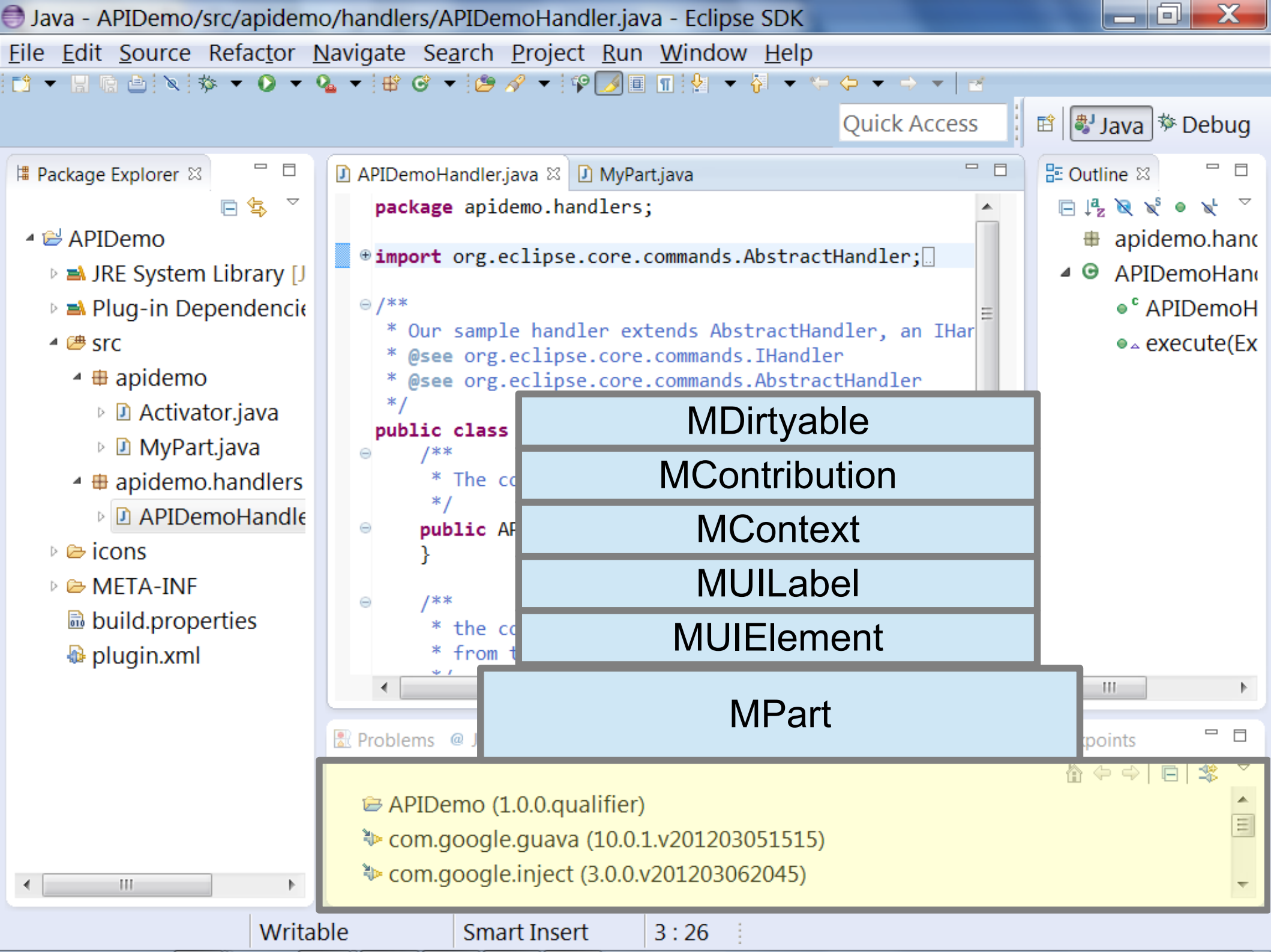

Java - APIDemo/src/apidemo/handlers/APIDemoHandler.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
APIDemo
JRE System Library [J
Plug-in Dependencie
src
apidemo
Activator.java
MyPart.java
apidemo.handlers
APIDemoHandle
icons
META-INF
build.properties
plugin.xml
APIDemoHandler.java
MyPart.java
package apidemo.handlers;
import org.eclipse.core.commands.AbstractHandler;
/**
MTrimmedWindow
public class APIDemoHandler extends AbstractHandler {
/**
* The constructor.
*/
public APIDemoHandler() {
}
/**
* the command has been executed, so extract extra
* from the application context.
Outline
apidemo.hanc
APIDemoHanc
APIDemoH
execute(Ex
Problems
Javadoc
Declaration
Plug-ins
Variables
Debug
Breakpoints
APIDemo (1.0.0.qualifier)
com.google.guava (10.0.1.v201203051515)
com.google.inject (3.0.0.v201203062045)
Writable
Smart Insert
3 : 26


Java - APIDemo/src/apidemo/handlers/APIDemoHandler.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
APIDemo
JRE System Library [J
Plug-in Dependencie
src
apidemo
Activator.java
MyPart.java
apidemo.handlers
APIDemoHandle
icons
META-INF
build.properties
plugin.xml
APIDemoHandler.java
MyPart.java
package apidemo.handlers;
import org.eclipse.core.commands.AbstractHandler;
/**
* Our sample handler extends AbstractHandler, an IHar
* @see org.eclipse.core.commands.IHandler
* @see org.eclipse.core.commands.AbstractHandler
*/
public class APIDemoHandler extends AbstractHandler {
/**
}
/**
* the command has been executed, so extract extra
* from the application context.
MTrimBar(s)
Outline
apidemo.hanc
APIDemoHanc
APIDemoH
execute(Ex
Problems
Javadoc
Declaration
Plug-ins
Variables
Debug
Breakpoints
APIDemo (1.0.0.qualifier)
com.google.guava (10.0.1.v201203051515)
com.google.inject (3.0.0.v201203062045)
Writable
Smart Insert
3 : 26


Java - APIDemo/src/apidemo/handlers/APIDemoHandler.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
MToolControl(s)
MToolbar(s)
Package Explorer
APIDemo
JRE System Library [J
Plug-in Dependencies
src
apidemo
Activator.java
MyPart.java
apidemo.handlers
APIDemoHandler
icons
META-INF
build.properties
plugin.xml
APIDemoHandler.java
MyPart.java
package apidemo.handlers;
import org.eclipse.core.commands.AbstractHandler;
/**
* Our sample handler extends AbstractHandler
* @see org.eclipse.core.commands.IHandler
* @see org.eclipse.core.commands.AbstractHandler
AbstractHandler {
* The constructor.
*/
public APIDemoHandler() {
}
/**
* the command has been executed, so extract extra
* from the application context.
Outline
Problems
Javadoc
Declaration
Plug-ins
Variables
Debug
Breakpoints
APIDemo (1.0.0.qualifier)
com.google.guava (10.0.1.v201203051515)
com.google.inject (3.0.0.v201203062045)
Writable
Smart Insert
3 : 26


Java - APIDemo/src/apidemo/handlers/APIDemoHandler.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
APIDemo
JRE System Library [J
Plug-in Dependencie
src
apidemo
Activator.java
MyPart.java
apidemo.handlers
APIDemoHandle
icons
META-INF
build.properties
plugin.xml
APIDemoHandler.java
MyPart.java
package apidemo.handlers;
import org.eclipse.core.commands.AbstractHandler;
/**
* Our sample handler extends AbstractHandler, an IHar
* @see org.eclipse.core.commands.IHandler
* @see org.eclipse.core.commands.AbstractHandler
*/
public class APIDemoHandler extends AbstractHandler {
/**
* The constructor.
*/
public APIDemoHandler() {
}
/**
* the command has been executed, so extract extra
* from the application context.
Outline
apidemo.han
APIDemoHan
APIDemoH
execute(Ex
MPartSashContainer
Problems
APIDemo (1.0.0.qualifier)
com.google.guava (10.0.1.v201203051515)
com.google.inject (3.0.0.v201203062045)
Writable
Smart Insert
3 : 26


Java - APIDemo/src/apidemo/handlers/APIDemoHandler.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
APIDemo
JRE System Library [J
Plug-in Dependencie
src
apidemo
Activator.java
MyPart.java
apidemo.handlers
APIDemoHandle
icons
META-INF
build.properties
plugin.xml
APIDemoHandler.java
MyPart.java
package apidemo.handlers;
import org.eclipse.core.commands.AbstractHandler;
/**
* Our sample handler extends AbstractHandler, an IHar
* @see org.eclipse.core.commands.IHandler
* @see org.eclipse.core.commands.AbstractHandler
*/
public class APIDemoHandler extends AbstractHandler {
/**
* The constructor.
*/
public APIDemoHandler() {
}
/**
* th
* fr
Outline
apidemo.hanc
APIDemoHan
APIDemoH
execute(Ex
MPartStack
Problems
Javadoc
Declaration
Plug-ins
Variables
Debug
Breakpoints
APIDemo (1.0.0.qualifier)
com.google.guava (10.0.1.v201203051515)
com.google.inject (3.0.0.v201203062045)
Writable
Smart Insert
3 : 26


Java - APIDemo/src/apidemo/handlers/APIDemoHandler.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
APIDemo
JRE System Library [J
Plug-in Dependencie
src
apidemo
Activator.java
MyPart.java
apidemo.handlers
APIDemoHandle
icons
META-INF
build.properties
plugin.xml
APIDemoHandler.java
MyPart.java
package apidemo.handlers;
import org.eclipse.core.commands.AbstractHandler;
/**
* Our sample handler extends AbstractHandler, an IHar
* @see org.eclipse.core.commands.IHandler
* @see org.eclipse.core.commands.AbstractHandler
*/
public class
Outline
apidemo.hanc
APIDemoHan
APIDemoH
execute(Ex
MDirtyable
MContribution
MContext
MUILabel
MUIElement
MPart
Problems
APIDemo (1.0.0.qualifier)
com.google.guava (10.0.1.v201203051515)
com.google.inject (3.0.0.v201203062045)
Writable
Smart Insert
3 : 26

Simple Scenario Demo

- Standard (3.x) Command to add a new view to the 'bottom' stack
  - Uses new API to manipulate the model
  - New MyPart is an e4 part
    - It's a POJO
    - Gets its info through DI
    - Listens to both model and preference changes

What's Next ?

- Docs...Docs...Docs !!
- Loosen up extension point 'class' restrictions
- Finish up LifeCycle events
- Start blurring the Eclipse 4 / E4 boundary
  - Start breaking some Views and services away from depending on the Workbench

A Start on the Docs

- http://wiki.eclipse.org/Eclipse4/API
- http://wiki.eclipse.org/Eclipse4/RCP/Dependency_Injection

Questions ?

A Start on the Docs

- http://wiki.eclipse.org/Eclipse4/API
- http://wiki.eclipse.org/Eclipse4/RCP/Dependency_Injection

```java
public class APIDemoHandler extends AbstractHandler {
    /**
     * The constructor.
     */
    public APIDemoHandler() {
    }

    /**
     * the command has been executed, so extract extract the needed information
     * from the application context.
     */
    public Object execute(ExecutionEvent event) throws ExecutionException {
        IWorkbenchWindow window = HandlerUtil.getActiveWorkbenchWindowChecked(event);
        MApplication theApp = (MApplication) window.getService(MApplication.class);
        if (theApp != null) {
            MWindow activeWin = theApp.getSelectedElement();
            EModelService ms = activeWin.getContext().get(EModelService.class);
            MPartStack btmStack = (MPartStack) ms.find("bottom", activeWin);
            MPart curPart = (MPart) ms.find("APIDemo.myPart", btmStack);
            if (curPart == null) {
                curPart = ms.createModelElement(MPart.class);
                curPart.setElementId("APIDemo.myPart");
                curPart.setCloseable(true);
                curPart.setLabel("API Demo");
                curPart.setContributionURI("bundleclass://APIDemo/apidemo.MyPart");
                btmStack.getChildren().add(curPart);
            }

            EPartService ps = activeWin.getContext().get(EPartService.class);
            ps.activate(curPart);
        }
        return null;
    }
}
```

```
@Focus void setFocus() {
    label.setFocus();
}

@Inject @Optional
void tbrHandler(@UIEventTopic(UIEvents.UIElement.TOPIC_WIDGET) Event eventData) {
    Object changedElement = eventData.getProperty(UIEvents.EventTags.ELEMENT);

    if (!(changedElement instanceof MPart))
        return;

    visCount = countVisibleElements();
    updateMessage();
}

@Inject
void animationsPrefChanged(@Preference(nodePath="org.eclipse.ui",
                                       value="SHOW_MEMORY_MONITOR") String value) {
    showHeapValue = value == null ? "false" : value;
    updateMessage();
}

@Inject
void activePartChanged(@Named(IServiceConstants.ACTIVE_PART) MPart newPart) {
    apLabel = newPart == null ? null : newPart.getLabel();
    updateMessage();
}

@Inject
void selChanged(@Optional @Named(IServiceConstants.ACTIVE_SELECTION) Object newSel) {
    selCount = 0;
    if (newSel instanceof IStructuredSelection) {
        selCount = ((IStructuredSelection)newSel).size();
    }
    updateMessage();
}
```