ECF :

"Easy" Communication Framework

a cura di Gerardo Lombardo [gerardo_lomb@libero.it]

ECF (**E**clipse **C**ommunication **F**ramework)

framework per la creazione di applicazioni distribuite (plugins, RCP appl., OSGi servers…) multi-protocollo su piattaforma Eclipse.

Easy perché…

Le API di comunicazione di ECF possiedono un altissimo livello di astrazione, quindi:

L'attenzione dello sviluppatore si sposta quasi esclusivamente sulla Business Logic e sul controllo della UI.

Due *prospettive* di utilizzo :

- **Utente**:
  strumenti di comunicazione multiprotocollo integrati in Eclipse (IRC, GoogleTalk, Shared Editor…), disponibili nella perspective *Communications*.

- **Sviluppatore**:
  insieme di API (open source) che permettono di integrare e/o estendere facilmente gli strumenti di cui sopra nei propri plugin.

Principali interfacce:

- Container:
  rappresenta l'accesso ad un contesto/canale di comunicazione in base al tipo di protocollo che si intende utilizzare.
  - rappresenta una sessione di comunicazione
  - sincrono/asincrono…
  - client/server, publish-and-subscribe…
- ID:
  identificatore relativo ad una istanza di container.
  - unico e immutabile all'interno del namespace in cui è stato generato…

Concetti preliminari:

- ContainerFactory:
  extension point che implementa l'interfaccia che genera il tipo di container specifico del protocollo di comunicazione scelto.
  `org.eclipse.ecf.containerFactory`
  `(org.eclipse.ecf.core.provider.IContainerInstantiator)`
- Namespace:
  extension point che definisce uno spazio degli ID per la creazione e la gestione dei container di uno stesso tipo.
  `org.eclipse.ecf.identity.Namespace`
  `(org.eclipse.ecf.core.identity.Namespace)`

Concetti preliminari:

- Provider:

  plugin che implementa i dettagli di un protocollo di comunicazione, dall'addressing alle modalità di connessione e trasferimento dati; o equivalentemente:

  Plugin che include almeno le due estensioni appena viste:

  - ContainerFactory (genera Container)
  - Namespace (addressing protocollo)

Concetti preliminari:

- Protocol Adapter Pattern:

  meccanismo tramite il quale si ottiene, al runtime, il ContainerAdapter specializzato (prodotto dalla ContainerFactory corrispondente) per il contesto/protocollo di comunicazione che si intende utilizzare .

  - l'interfaccia IContainer estende IAdaptable:
    `IContainer.getAdapter(<interface>)`

  - ad esempio:

```
container.getAdapter(IPresenceContainer.class);
```

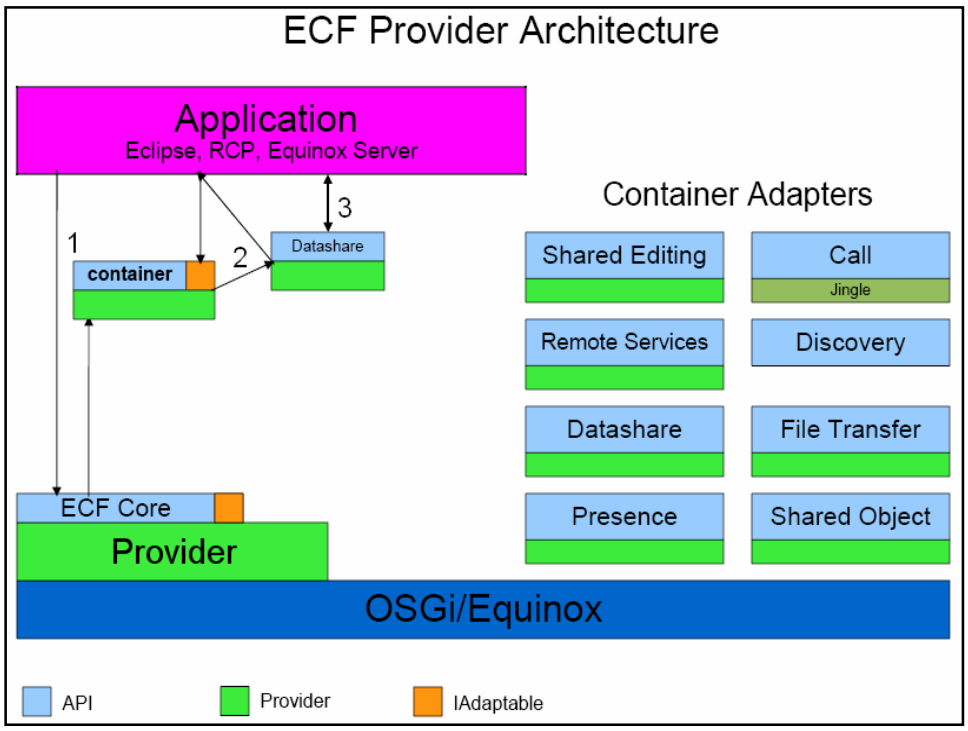
Ciclo di vita di un Container :

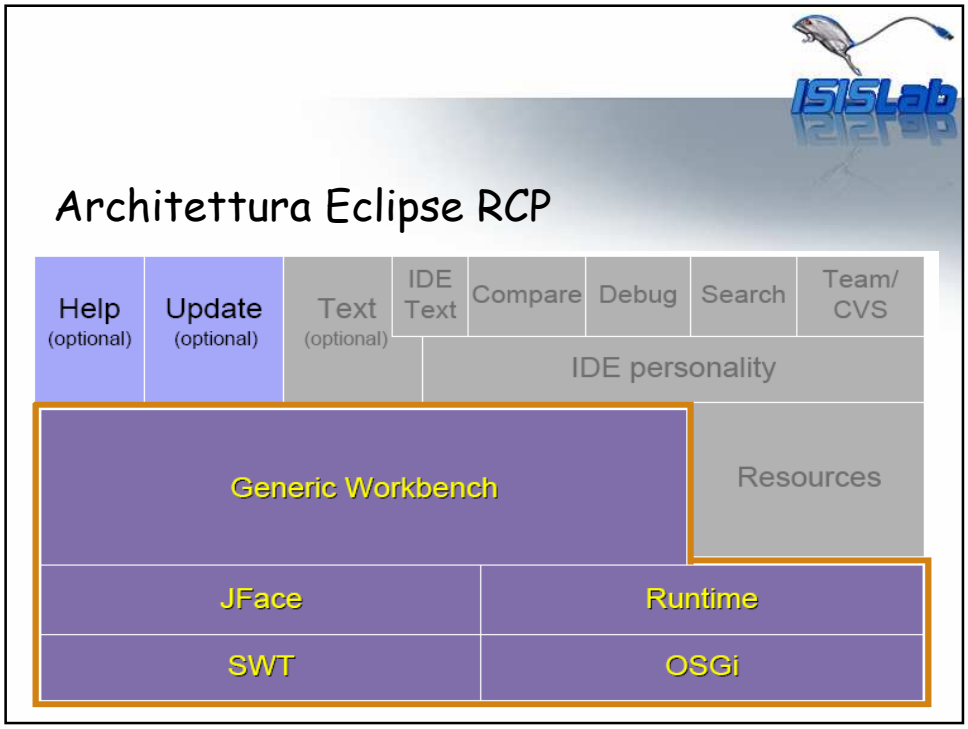
- Creazione (tramite Containerfactory)
- Setup (tramite Adapter …)
- Connessione (`container.connect(namespace,ID)` …)
- Comunicazione (…)
- Disconnessione

```
// Creazione Container
IContainer container =
  ContainerFactory.getDefault().
   createContainer("ecf.xmpp.smack");

// Creazione ID...
Namespace ns =
  container.getConnectNamespace();

ID containerID =
 IDFactory.getDefault().
  createID(ns,"nomeaccount@gmail.com");
```

```
// Setup per il protocollo xmpp...
IConnectContext connectContext =
 ConnectContextFactory
  .createPasswordConnectContext("password");
 ...
// Adapter per invio/ricez. Messaggi Chat...
IPresenceContainerAdapter
presenceContainerAdapter =
  (IPresenceContainerAdapter) container
 .getAdapter(IPresenceContainerAdapter.class);

if (presenceContainerAdapter != null){

IChatManager icm =
  presenceContainerAdapter.getChatManager();

icm.addMessageListener(messageListener);
```

```
 ...
// Connessione Container
container.connect(containerID,connectContext);

 ...

// Comunicazione...
IChatMessageSender icms =
  icm.getChatMessageSender();

icms.sendChatMessage(remoteID, "Ciao!");

 ...
container.disconnect();
 ...
container.dispose();
```


ECF Provider Architecture
Application
Eclipse, RCP, Equinox Server
1
2
3
container
Datashare
Container Adapters
Shared Editing
Call
Jingle
Remote Services
Discovery
Datashare
File Transfer
Presence
Shared Object
ECF Core
Provider
OSGi/Equinox
API
Provider
IAdaptable


Architettura Eclipse RCP
Help
(optional)
Update
(optional)
Text
(optional)
IDE
Text
Compare
Debug
Search
Team/
CVS
IDE personality
Generic Workbench
Resources
JFace
Runtime
SWT
OSGi

I Provider concretizzano i modelli di comunicazione definiti astrattamente nelle API di ECF.

Seguono ora le descrizioni delle principali API con i corrispondenti Provider riconosciuti ufficialmente dal progetto ECF…

Principali API e corrispondenti Provider

ECF Core `[org.eclipse.ecf]`

Definisce le interfacce IContainer e ID, con i corrispondenti extension point visti prima, cioè ContainerFactory e Namespace.

Include inoltre la fase di avvio del framework core.

Datashare `[org.eclipse.ecf.datashare]`

Supporto per la definizione e creazione di canali (IChannel) di comunicazione per lo scambio di messaggi asincrono tramite l'API IChannelContainer. I canali di comunicazione possono essere di tipo point-to-point oppure publish/subscribe.

Principali API e corrispondenti Provider

Shared Object `[org.eclipse.ecf.sharedobject]`

L'interfaccia ISharedObjectContainer è capace di creare, replicare e gestire oggetti (ISharedObject) in contesti distribuiti (ad es. Gruppi).

Supporto per invio/ricez. Messaggi asincrono su di un servizio/protocollo di comunicazione arbitrario.

- i provider XMPP e RemoteService utilizzano SharedObj

Remote Service `[org.eclipse.ecf.datashare]`

Creazione, registrazione e invocazione di servizi (IRemoteService) tramite l'interfaccia IRemoteServiceContainer.

Principali API e corrispondenti Provider

Discovery `[org.eclipse.ecf.discovery]`

Scoperta di servizi di rete con l'interfaccia IDiscoveryContainer.

- Zeroconf/Rendevous
- jSLP (Java Service Location Protocol)

File Transfer `[org.eclipse.ecf.filetransfer]`

Invio/Ricezione File tramite le interfacce IOutgoingFileTransferContainer e IRetrieveFileTransferContainer.

- Apache Httpclient su HTTP/HTTPS (file retrieval)
- BitTorrent (file retrieval)
- XMPP (peer-to-peer file sending)

Principali API e corrispondenti Provider

Presence `[org.eclipse.ecf.presence]`

Gestione presenza, invio/ricez. Messaggi e altro su IM Chat.

- IRC (Internet Relay Chat)
- MSN (Microsoft Network)
- XMPP/Jabber (GoogleTalk, …)
- Yahoo IM

Call `[org.eclipse.ecf.call]`

Gestione presenza e conversazioni su VOIP.

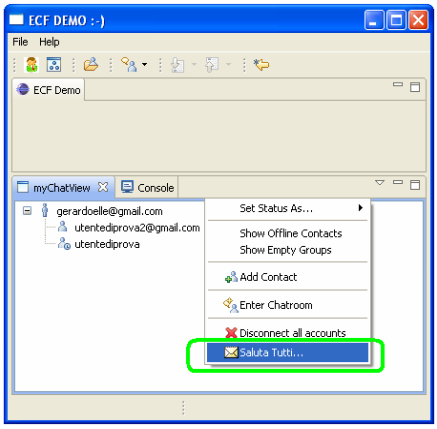
- Skype (basato su Skype4Java, richiede Skype in esecuzione)
- Jingle (basato su XMPP, richiede Java-Media-Framework)

ECF Demo

Semplice applicazione Eclipse RCP che integra:

- un wizard di connessione (Connectwizard) al provider XMPP(ecf.xmpp.smack), compatibile con GoogleTalk.
- Aggiunta della semplice funzione helloBroadcast alla ActionBars della View per la chat (MultiRosterView).

In poche parole viene aggiunta una nuova funzione nella finestra della chat attivabile tramite popup menu:



La funzione helloBroadcast invia a tutti i contatti in linea(testato tra contatti GoogleTalk) un messaggio di saluto.

La maggior parte del codice riguarda:

- Interfaccia grafica del wizard
- L'estensione della classe MultiRosterView (**`org.eclipse.ecf.presence.ui`**) aggiungendo una nuova IAction(JFace)

Del ConnectWizard verrà spiegato solo il funzionamento poiché la maggior parte del codice riguarda la UI.

Quindi sarà illustrata in dettaglio l'implementazione della Chat View che implementa l'azione (IAction) helloBroadcast.

```
<extension point  =
  "org.eclipse.ecf.ui.connectWizards">
    <wizard class =
      "EcfTests.ConnectWizard"
       containerFactoryName =
        "ecf.xmpp.smack"

       id="ECF_DEMO.ConnectWizard“

       name="Wizard Esempio">
  </wizard>
</extension>
```

Un Wizard è formato da una o più WizardPage, il ConnectWizard dopo aver ricevuto i parametri di connessione dalla ConnectWizardPage:

- crea un Listener per gestire l'evento di connessione
- effettua la connessione del container

```
public class ConnectWizard extends
 Wizard implements IConnectWizard {

       ConnectWizardPage page;

       ...
```

Ovviamente il Listener attiva la Chat View e vi aggiunge il container connesso:

```
private void openView()
 {
   myChatView view =
    (myChatView)  workbench.
    getActiveWorkbenchWindow()
    .getActivePage()
    .showView(myChatView.VIEW_ID);

  view.addContainer(container);
 }
```

Seguono infine l'implementazione dell'estensione della classe MultiRosterView che include la definizione della IAction(JFace) che effettua il broadcast del messaggio di saluto.

```
<view
      class="EcfTests.myChatView"
      id="ECF_DEMO.myChatView"
      name="myChatView">
</view>
```

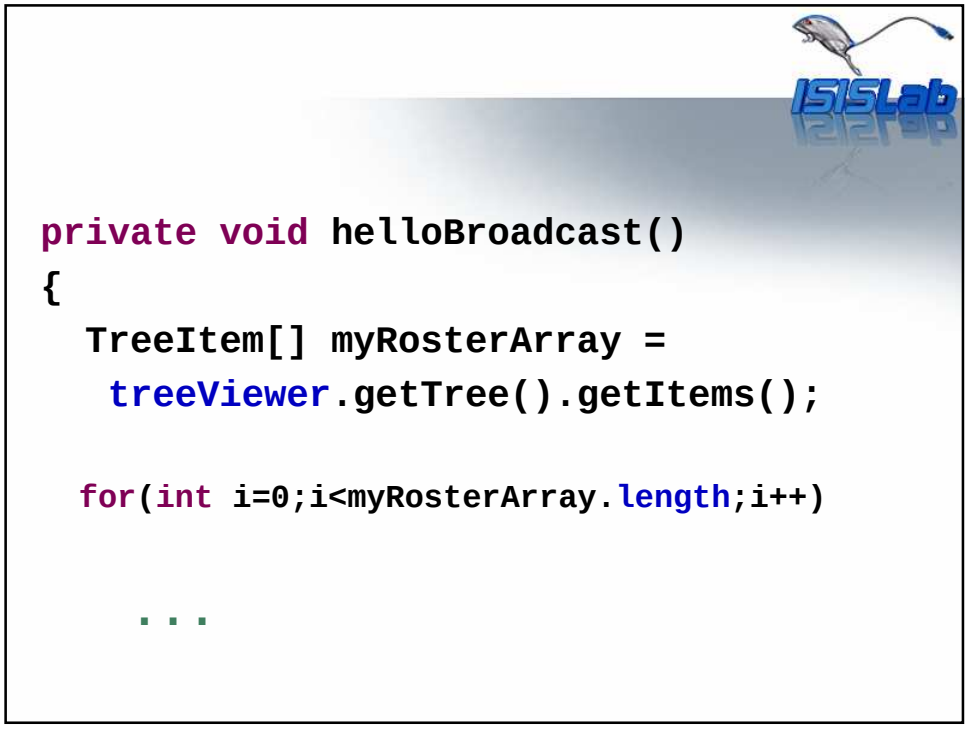
```
public class myChatView extends
  MultiRosterView
{
 public static final String VIEW_ID
  = "ECF_DEMO.myChatView";

 IAction myAction;
```

```
public void createPartControl(Composite
  parent)
{
  super.createPartControl(parent);

  makeMyAction();
  // Prepara la logica della Action...

  addMyAction();
  // Registra la Action nella ActionBar
}
```

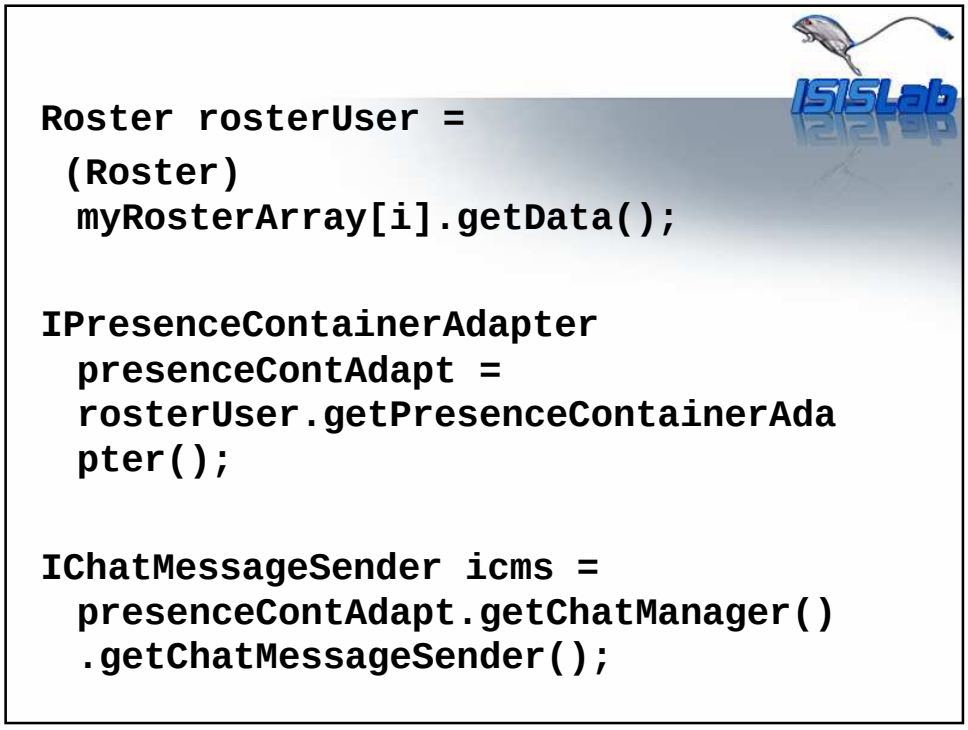
```
private void makeMyAction(){

myAction = new Action() {
   public void run(){
      helloBroadcast();
   }
 };

myAction.setImageDescriptor(...);
myAction.setText("Saluta Tutti...");
```

```
treeViewer.getTree().addPaintListener
  (new PaintListener()
{
 public void paintControl(PaintEvent e)
 {
  //Controlla se c'è almeno un
  contatto in linea
  if (isEmptyTree())
    myAction.setEnabled(false);
  else
    myAction.setEnabled(true);
}});
```

```
private void addMyAction()
{
 IActionBars actionBar =
  getViewSite().getActionBars();

 IMenuManager menuBarManager =
  actionBar.getMenuManager();

 menuBarManager.add(myAction);
}
```



```
private void helloBroadcast()
{
  TreeItem[] myRosterArray =
   treeViewer.getTree().getItems();

  for(int i=0;i<myRosterArray.length;i++)

    ...
```



```
Roster rosterUser =
 (Roster)
  myRosterArray[i].getData();

IPresenceContainerAdapter
  presenceContAdapt =
  rosterUser.getPresenceContainerAda
  pter();

IChatMessageSender icms =
  presenceContAdapt.getChatManager()
  .getChatMessageSender();
```

```
//Scorre la lista dei contatti di un
  rosterUser
while (it.hasNext())
{
 RosterEntry entry=(RosterEntry)it.next();
 ID targetID = entry.getUser().getID();

 Type statusType =
  entry.getPresence().getType();

if(statusType!=IPresence.Type.UNAVAILABLE)
{
 icms.sendChatMessage(targetID, "Ciao a
  tutti!");
}}
```

Riferimenti ECF:

http://www.eclipse.org/ecf/

http://wiki.eclipse.org/Eclipse_Communication_Framework_Project

http://wiki.eclipse.org/ECF_API_Docs

http://wiki.eclipse.org/ECF_Providers

http://wiki.eclipse.org/Category:Eclipse_Communication_Framework

http://ecf1.osuosl.org/