Injeqt - dependency injection framework for Qt applications

Few words about me

Rafał Malinowski
www.kadu.im
C++/Qt consultant
www.rkvalue.com

Lets look at some code

```
void YourAccounts::accountCreated(Account account)
{
    if (!account)
        return;

    AccountManager::instance()->addItem(account);
    account.accountContact().setOwnerBuddy(Core::instance()->myself());

    ConfigurationManager::instance()->flush();
    selectAccount(account);
}
```

Lets look at some code

```
void YourAccounts::accountCreated(Account account)
{
    if (!account)
        return;

    AccountManager::instance()->addItem(account);
    account.accountContact().setOwnerBuddy(Core::instance()->myself());

    ConfigurationManager::instance()->flush();
    selectAccount(account);
}
```

Dependency injection

Pass what is required to object
All dependencies are explicit
All dependencies are mockable

A bit better

```
void YourAccounts::setAccountManager(AccountManager *x)
{
    m_accountManager = x;
}

void YourAccounts::setMyself(Buddy x)
{
    m_myself = x;
}

void YourAccounts::setConfigurationStorage(ConfigurationStorage *x)
{
    m_configurationStorage = x;
}
```

A bit better

```
YourAccounts::YourAccounts(AccountManager *x, Buddy y, ConfigurationStorage *z)
    : m_accountManager{x}, m_myself{y}, m_configurationStorage{z}
{
    ...
}
```

A bit better

```
void YourAccounts::accountCreated(Account account)
{
    if (!account)
        return;

    m_accountManager->addItem(account);
    account.accountContact().setOwnerBuddy(m_myself);

    m_configurationStorage->flush();
    selectAccount(account);
}
```

Core class

```
m_pluginActivationService = new PluginActivationService{this};
m_pluginDependencyHandler = new PluginDependencyHandler{this};
m_pluginStateService = new PluginStateService{this};

m_pluginManager = new PluginManager{this};
m_pluginManager->setPluginActivationService{m_pluginActivationService};
m_pluginManager->setPluginDependencyHandler{m_pluginDependencyHandler};
m_pluginManager->setPluginStateService{m_pluginStateService};
```

Automate it!

Manual creation and wiring of objects?
Compiler can do it for us!

Frameworks

C# – Spring.NET
Java – Spring, Guice
C++ – no reflection :(

Qt/MOC

Proof of concept
C++11/C++14
Injeqt!

It works!

Will be released before end of the year

Kadu master already uses it

What works?

```
class KADUAPI ConfigurationWriter final : public QObject
{
    Q_OBJECT
public:
    Q_INVOKABLE explicit ConfigurationWriter();
...
private slots:
    INJEQT_SETTER void setConfiguration(Configuration *configuration);
    INJEQT_SETTER void setConfigurationPathProvider(
        ConfigurationPathProvider *configurationPathProvider);
};
```

What works?

```
class KADUAPI ConfigurationFactory final : public QObject
{
    Q_OBJECT
public:
    Q_INVOKABLE explicit ConfigurationFactory(QObject *parent = nullptr);
    Q_INVOKABLE Configuration * createConfiguration() const;
...
private slots:
    INJEQT_SETTER void setConfigurationPathProvider(
        ConfigurationPathProvider *configurationPathProvider);
};
```

What works?

```
class KADUAPI ConfigurationModule : public injeqt::module
{
public:
    ConfigurationModule()
    {
        add_type<ConfigurationFactory>();
        add_type<ConfigurationPathProvider>();
        add_type<ConfigurationWriter>();
        add_factory<Configuration, ConfigurationFactory>();
    }
};
```

What works?

```
class KADUAPI CoreModule : public injeqt::module
{
public:
    explicit CoreModule(QString profileDirectory)
    {
        m_pathsProvider = make_not_owned<PathsProvider>(
            std::move(profileDirectory));
        add_ready_object<PathsProvider>(m_pathsProvider.get());
    }
    virtual CoreModule() {}
private:
    not_owned_qptr<PathsProvider> m_pathsProvider;
};
```

What works?

```
auto modules = std::vector<std::unique_ptr<injeqt::module>>{};
modules.emplace_back(make_unique<ChatWidgetModule>());
modules.emplace_back(make_unique<CoreModule>(std::move(profileDirectory)));
modules.emplace_back(make_unique<ConfigurationModule>());

auto injector = injeqt::injector{std::move(modules)};

auto configurationWriter = injector.get<ConfigurationWriter>();
auto application = injector.get<Application>();
```

Release

Tests
Build
Documentation
Examples
Comments

Future

Add scopes and subinjectors
Add constructor dependencies
Add signal/slots autoconnection
C++ reflection (202x?)

Links

github.com/vogel/injeqt
rkvalue.com
www.kadu.im
blog.kadu.im

Thank you!

Don't forget to vote!
My name is: Rafał Malinowski