Arquitetura MVC - Model-View-Controller

Introdução

O padrão MVC (Model-View-Controller) é um padrão arquitetural para a engenharia de software. O padrão permite que o programador possa separar a interface do usuário (camada de apresentação), os dados do aplicativo e lógica do negócio tornando muito mais fácil para manter o código e acomodar as mudanças necessárias.

- Model - O modelo é uma representação de dados utilizado especificamente em um aplicativo.
- View - A interface de um aplicativo. Ela torna o modelo de exposição ou de interação.
- Controller - Manipula eventos de interação com o usuário e, normalmente, invoca as mudanças no modelo.

PureMVC

PureMVC é uma fonte livre, aberta, framework leve para criação de aplicativos baseados na Model-View-Contoller. Ele é suportado por muitas linguagens, incluindo Java e pode ser implementado em um aplicativo Java ME. Visite http://www.puremvc.org para aprender sobre a estrutura, melhores práticas e para baixar a versão mais recente. PureMVC também está disponível para Python.

Usando PureMVC com Java ME

PureMVC é usado para construir a base de um pedido em forma bem estruturada. Isso permite que o programador facilmente tenham os aplicativos que eles estão construindo para muitos dispositivos e plataformas. Também torna mais fácil de manter, ampliar e alterar o projeto.

Download the example Login application.

Este exemplo pode ser utilizado como base em qualquer projeto Java Mobile. Atualmente, o aplicativo usa a versão 0.2 do Java portátil. A PureMVC é uma rápida visão geral de como alguns quadros operam com o Login demonstrativo. Para obter uma plena compreensão do PureMVC visite o site e estude a muito a documentação. Os exemplos estão escritos em ActionScript 3, mas a teoria se aplica a todas as linguagens suportadas.

O framework puremvc trabalha com 4 fatores.

- O Modelo '(Model)'
- A Visualização '(View)'
- O Controle '(Controller)'
- A Fachada '(Facade)'

Todos os pedidos de cemeço para a fachada ApplicationFacade. É o ponto central de comunicação para o framework.

```
private ApplicationFacade facade = ApplicationFacade.getInst();
...
    protected void startApp()throws MIDletStateChangeException
    {
  this.facade.startup(this);
    }
...
```

ApplicationFacade.java

```
package org.puremvc.java.demos.j2me.login;
import org.puremvc.java.patterns.facade.Facade;
import org.puremvc.java.patterns.observer.Notification;
import org.puremvc.java.demos.j2me.login.controller.StartupCommand;
import org.puremvc.java.demos.j2me.login.controller.ProcessLogin;
import org.puremvc.java.demos.j2me.login.LoginExample;
public class ApplicationFacade extends Facade
{
```


(C) Copyright Nokia 2014. All rights reserved.

```
  public static final String STARTUP = "startup";
  public static final String LOGIN = "login";
  public static final String SUBMIT_LOGIN = "submitLogin";
  public static final String LOGIN_SUCCESSFUL = "loginSuccessful";
  public static final String LOGIN_FAIL = "loginFail";
  public static final String MAIN = "main";

  //Startup command notifications
      public static final String PREP_MODEL = "prepModel";
      public static final String PREP_VIEW = "prepView";

  private static ApplicationFacade instance = null;

  public static LoginExample midlet;

  public static ApplicationFacade getInst()
  {
   if(instance == null)
   {
    instance = new ApplicationFacade();
   }

   return instance;
  }

  protected void initializeController()
  {
   super.initializeController();

   registerCommand(STARTUP, StartupCommand.class);
   registerCommand(SUBMIT_LOGIN, ProcessLogin.class);
  }

  public void startup(LoginExample midlet)
  {
   this.midlet = midlet;
   notifyObservers(new Notification(STARTUP, null, null));
  }
 }
```

Comunicação através do framework, registrando comandos, mediadores e proxies.

ApplicationFacade.java

```
registerCommand(SUBMIT_LOGIN, ProcessLogin.class);
```

PrepViewCommand.java

```
this.facade.registerMediator(new LoginScreenMediator());
```

PrepModelCommand.java

```
this.facade.registerProxy(new ItemDataProxy());
```

Cada tela de interface do usuário tem um mediador ligado a ele. O mediador interage com a interface do usuário e do framework.

(C) Copyright Nokia 2014. All rights reserved.

Mediadores respondem aos apelos por registar um interesse em uma notificação de chamadas e lidar com todas as chamadas através do método handleNotification.

LoginScreenMediator.java

```
public String[] listNotificationInterests()
{
 return new String[] {ApplicationFacade.LOGIN, ApplicationFacade.LOGIN_FAIL};
}



public void handleNotification(INotification note)
// As variáveis podem ser repassadas através de nota INotification.
{
 if(note.getName().equals(ApplicationFacade.LOGIN))
 {
  ApplicationFacade.midlet.fetchDisplay().setCurrent(getLoginScreen());
 }
 else if(note.getName().equals(ApplicationFacade.LOGIN_FAIL))
 {
  getLoginScreen().loginFail();

 }
}
```

Os mediadores também podem fazer uma chamada fora do framework usando facade.notifyObservers.

```
...
 this.facade.notifyObservers(new Notification(ApplicationFacade.SUBMIT_LOGIN, details,
 null));
```

Nota: Os detalhes são uma matriz de cadeia. Qualquer variável pode ser repassada através de uma notificação

Um controle de listens são chamados quando ele é registrado.

```
registerCommand(SUBMIT_LOGIN, ProcessLogin.class);
```

Os dados podem ser armazenados usando um objeto de valor (ItemDataVO.java) e acessado através de um proxy (ItemDataProxy.java)

Objetos de valor podem ser preenchidos e adicionados a um proxy, da seguinte forma.

ProcessLogin.java

```
//Adicionar alguns objetos ItemDataVO usando proxy
ItemDataProxy itemProxy = (ItemDataProxy)facade.retrieveProxy(ItemDataProxy.NAME);
for(int i=0; i<url.length;i++)
{
 ItemDataVO item = new ItemDataVO(url[i], data[i]);
 itemProxy.addItem(item);
}
```

Novamente, isto é apenas uma breve visão geral do framework PureMVC. http://www.puremvc.org Visite e leia a documentação para entender completamente o uso dele. Também o estudo sobre 'codebase demo Login'. O Demo é gratuito para você utilizar, alterar e criar em algo sobre ele. Por favor, ajude a comunidade, contribuindo com PureMVC de qualquer

(C) Copyright Nokia 2014. All rights reserved.

forma seja inovadora que melhore-o ou você pode conhecer a utilização deste framework com Java ME. Você também pode ajudar, fornecendo mais demos e tutoriais. Espero que você goste de criar uma melhor estrutura e mais fácil de manter as suas aplicações que utilizam o framework PureMVC.

Link interno

- Model-View-Controller application architecture

(C) Copyright Nokia 2014. All rights reserved.