新 Swing API で、簡単 GUI アプリケーション開発

花井 志生 IBM Japan
2008/4/30
http://www.ruimo.com/
ruimo@ruimo.com

mixi: ruimo
Twitter: ruimo

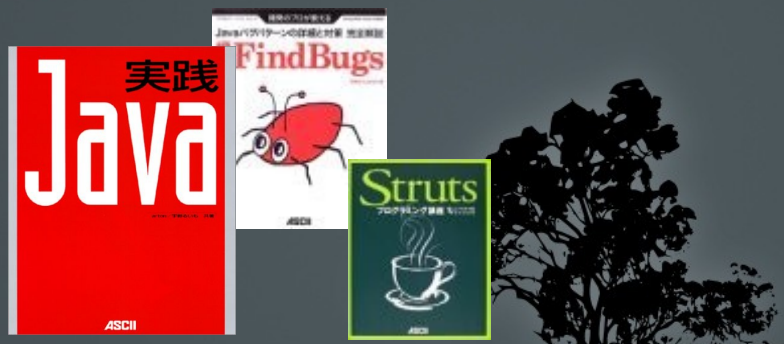
目次

- 自己紹介
- 今回取り上げる API
- 今回作成するプログラム
- 使用環境
- ビューの作成
- モデルの作成
- 入力チェック
- バリデータ
- 制御系も JavaBeans で！
- まとめ

今回の資料は、動画、サンプルを含めて、私のサイトで公開する予定です。
http://www.ruimo.com/publication/

自己紹介

- 主な仕事
  お客様向けカスタム SI
- Java との付き合い
  Java との出会いは 1.0.2 のころ。
  初の仕事: 2002 年 POS システムを Java で構築
  （初の Java の仕事が Swing(当時は JFC) アプリケーション)
- ペンネーム 宇野 るいも
  Java 関係の書籍執筆

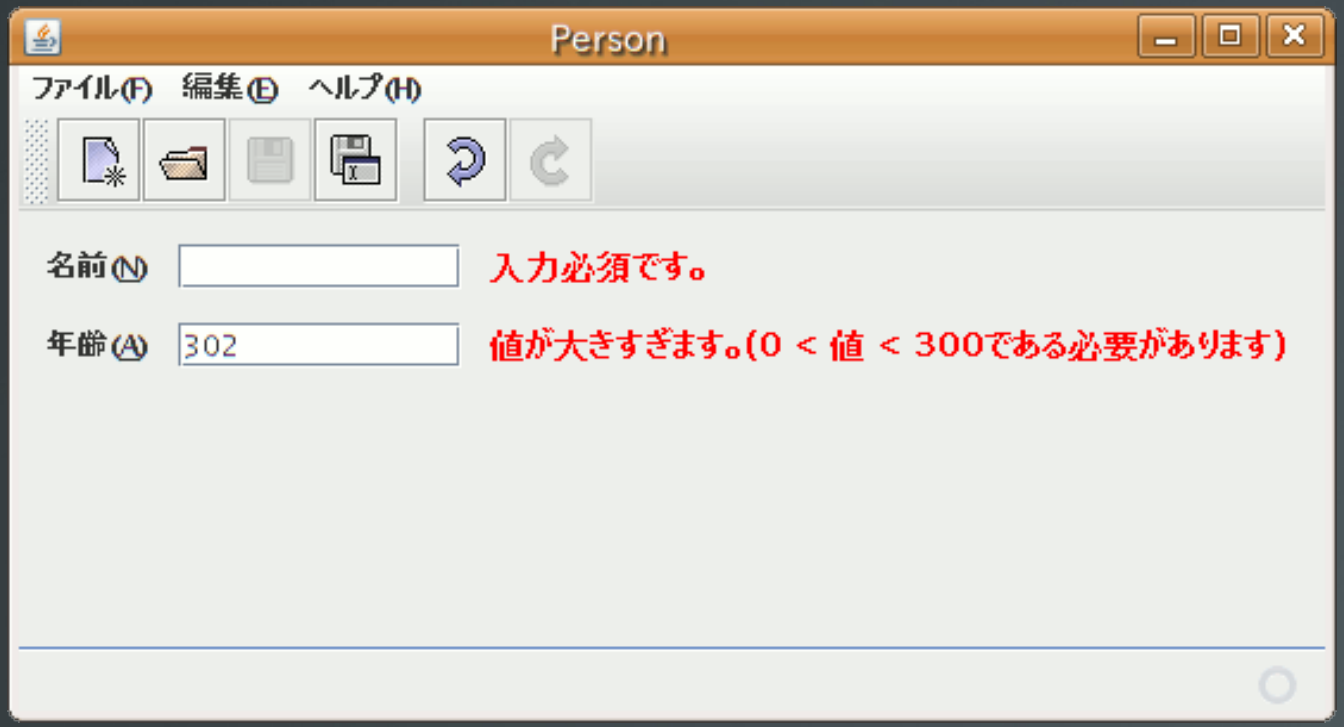


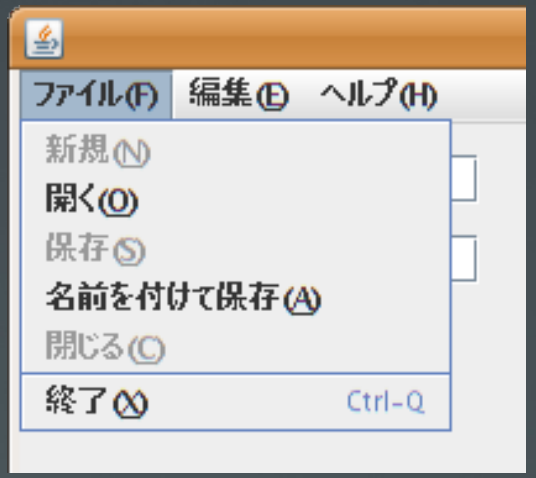
今回取り上げる API

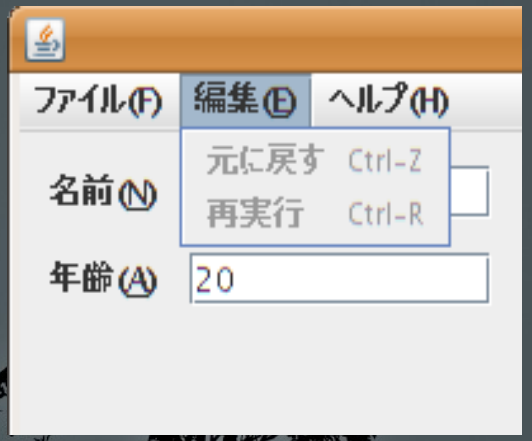
- JSR295 Beans Binding
  JavaBeans 同士をつなげるための API
  https://beansbinding.dev.java.net/
- JSR296 Swing Application Framework
  Swing アプリケーションを簡単に開発するための API
  https://appframework.dev.java.net/
- これらは、まだ検討中であり、将来変更される可能性もあるので、注意してください。

今回作成するプログラム

- 簡単な、名前と年齢を入力し、それをファイルに保存するアプリケーション。







使用している環境と準備

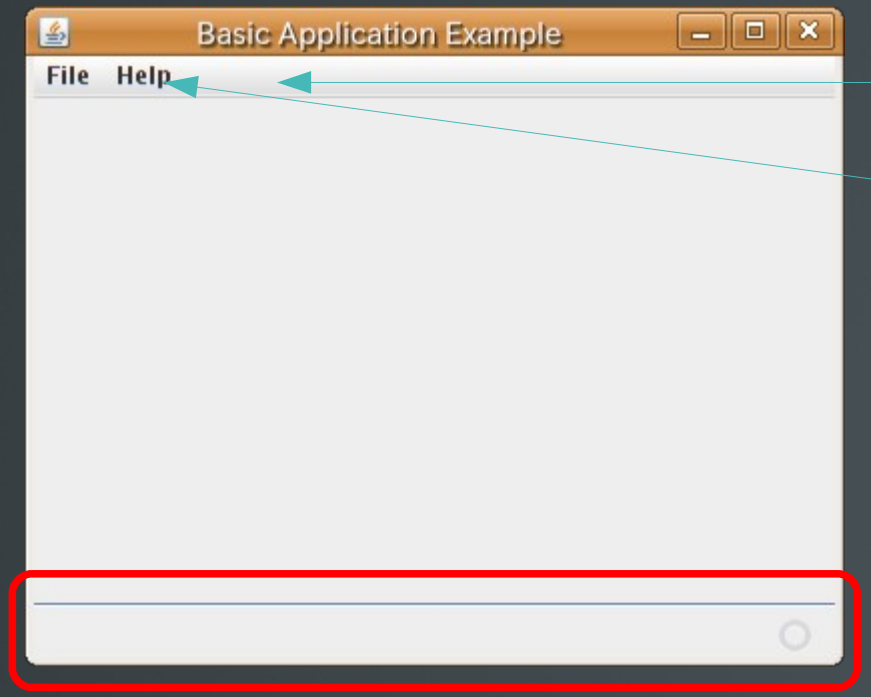
- Ubuntu 7.10
- JDK 1.6.0_04-b12 AMD64 Linux 版
- NetBeans 6.1 Beta (Build 200803050202)

- JSR295 1.2.1
- JSR296 1.03

ソースコードを登録しておきましょう。
Tools => Library Manager

プロジェクトの作成

- Java カテゴリから、Java Desktop Application を選ぶ。
- プロジェクトプロパティの Application を編集。



最低限のメニューが生成される

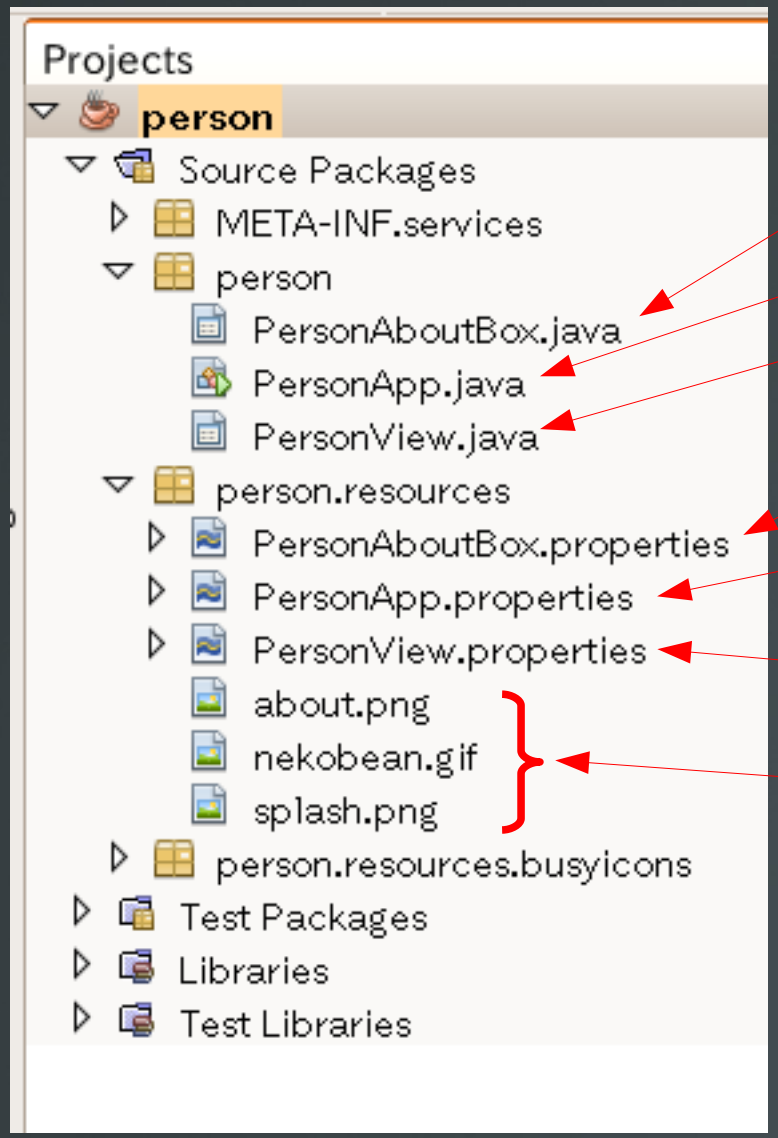
Help の中に About メニューが追加される
About ダイアログの表示

処理に時間を要する機能の実行時に
状況を表示するのに使用される。
SwingWorker

About ダイアログ

- プロジェクトプロパティの Appliction 項目に設定した値が自動的に反映される。
- それ以外は、resource パッケージ内のプロパティファイルで設定する。
- プロパティファイルには Add Locale で別言語のメッセージを追加できる。

自動生成されるアプリケーション構成



About ダイアログボックス

メインクラス

メインウィンドウ

About ダイアログボックスのプロパティファイル

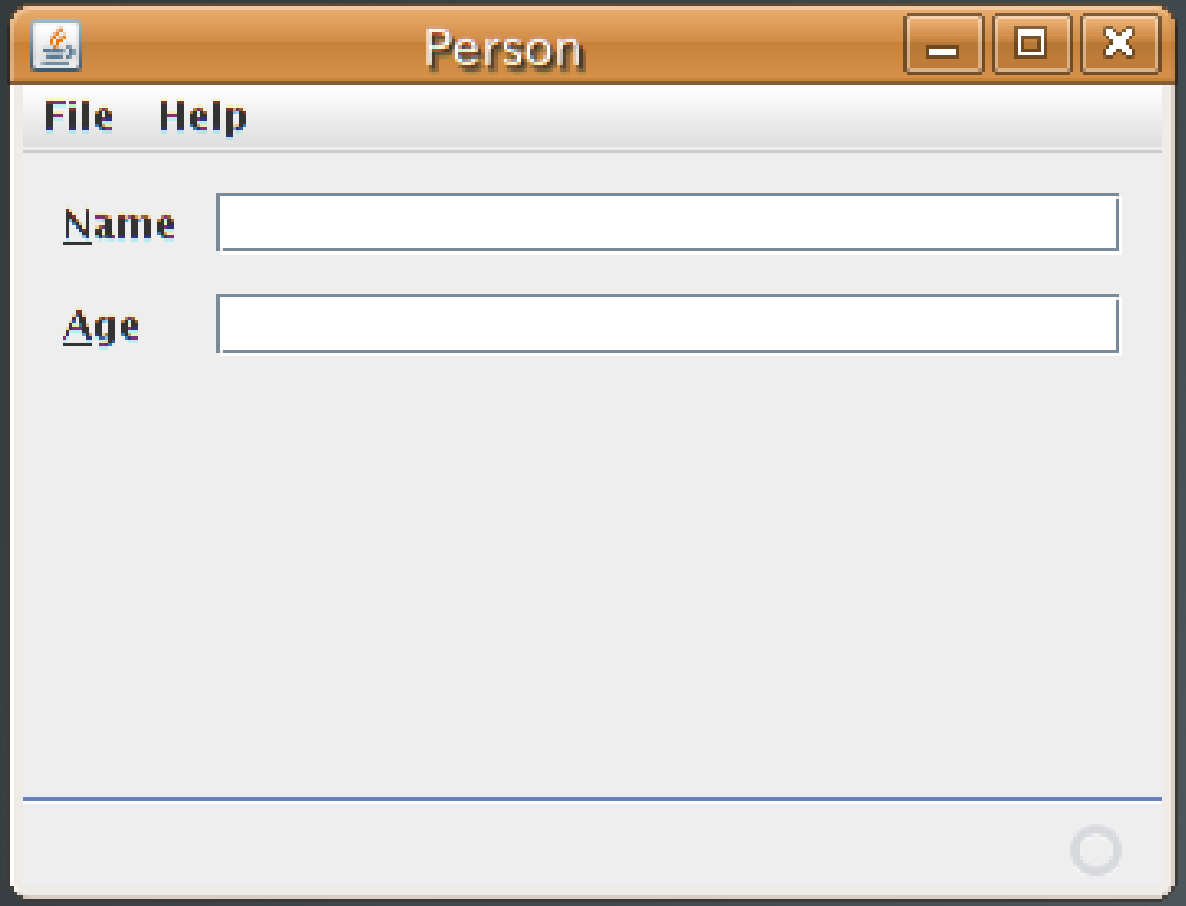
アプリケーションのプロパティファイル
プロジェクトプロパティで設定したもの

メインウィンドウのプロパティファイル

その他画像などのリソース

person

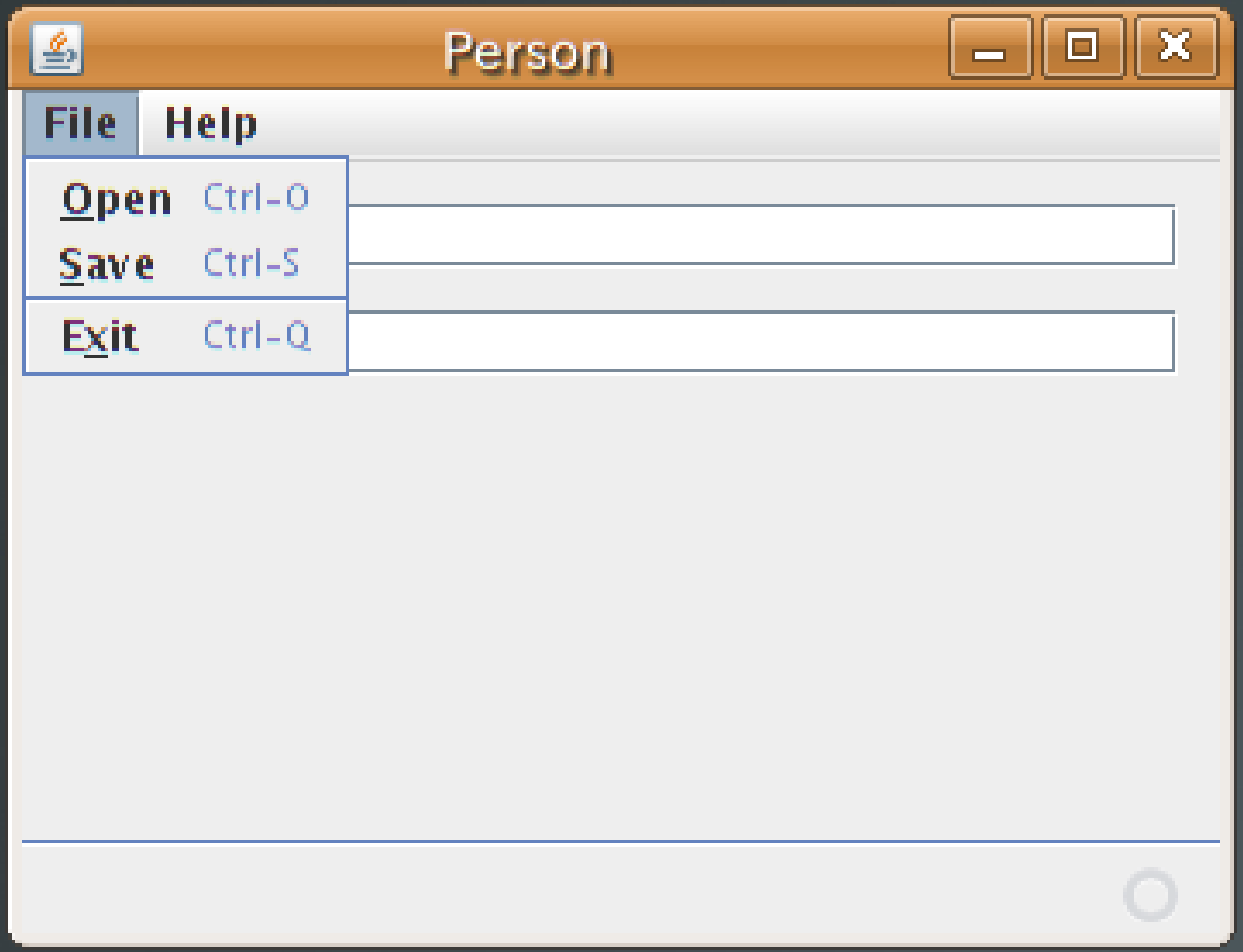
ビューの作成



person02

- 01-02.ogg

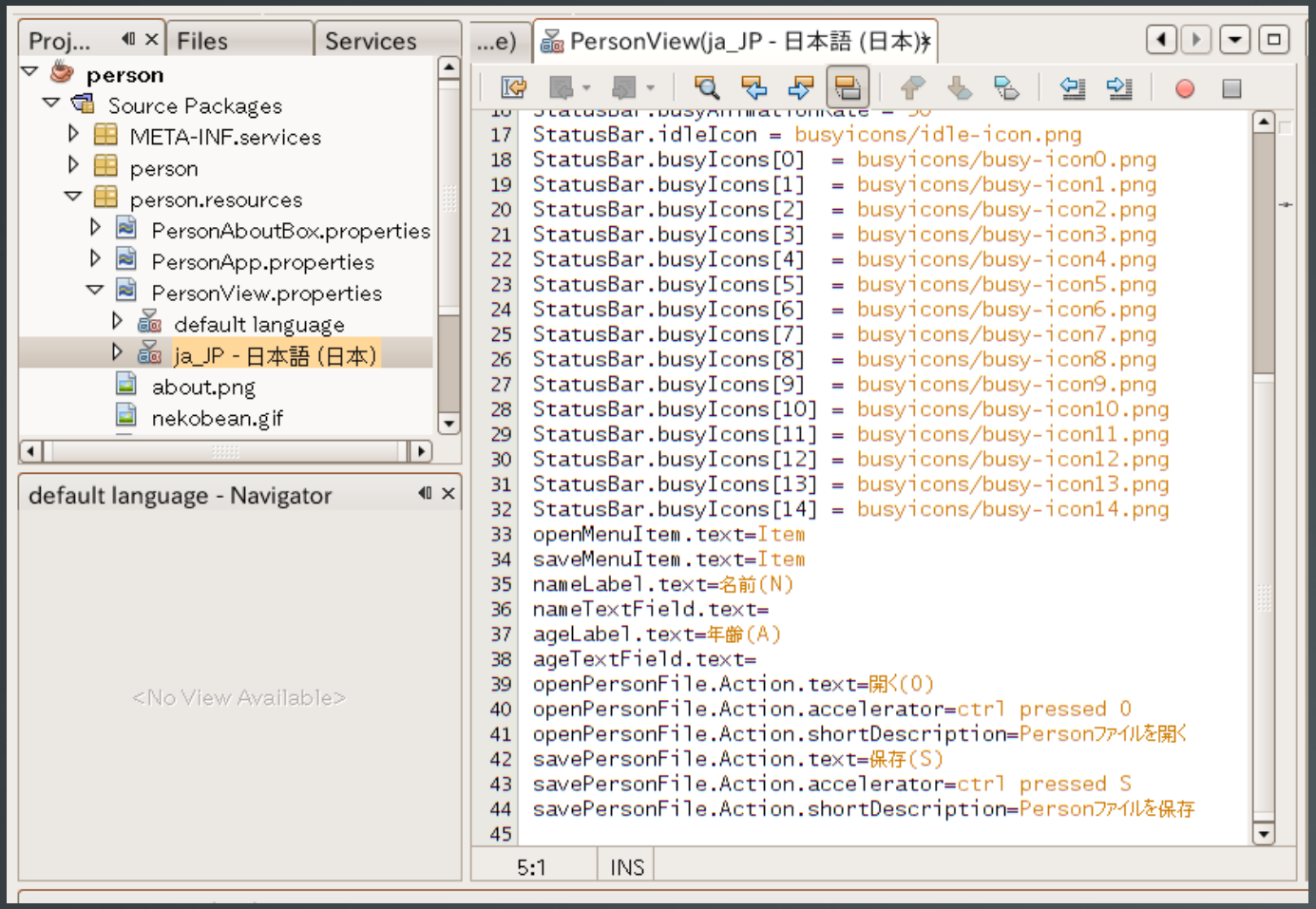
メニューを追加



- 02-03.ogg

person03

日本語プロパティファイルを用意

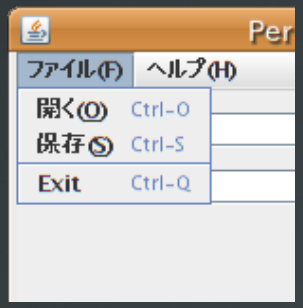


PersonView.properties に日本語ロカールを追加して、メッセージを日本語にする。

ロカールを追加した時点で、デフォルトロカールのメッセージ内容が、新ロカール用メッセージにコピーされる。

その後の、IDE によるメッセージ追加は、デフォルトロカールにしか追加されないので、自分で、日本語用のメッセージにコピーしないといけない点に注意。

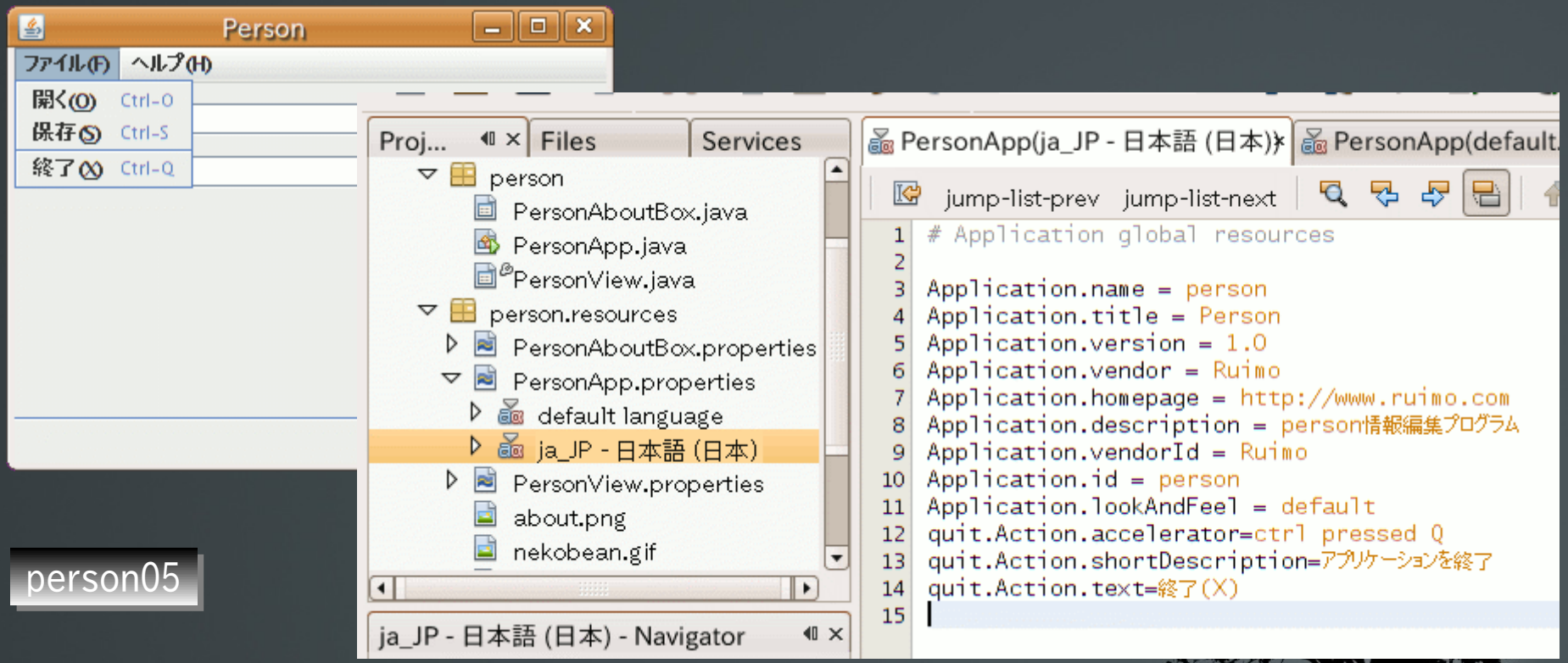
日本語プロパティファイルを用意



あれ？
Exit メニューは？

person04

Exit メニューは、PersonApp.properties に定義されています。



person05

モデルとビューをバインドする

- モデルをパレットに登録し
- パレットから、使いたいビューに登録
- UI コンポーネントにバインドする

- 05-06.ogg

モデルの読み込み

PersonViewのopenPersonFile()を実装する。

```
public void openPersonFile() {
    JFrame parent = getFrame();
    if (fileChooser.showOpenDialog(parent) != JFileChooser.APPROVE_OPTION) return;
    File file = fileChooser.getSelectedFile();
    ObjectInputStream ois = null;
    try {
        ois = new ObjectInputStream
            (new BufferedInputStream(new FileInputStream(file)));
        storePerson((Person)ois.readObject());
    }
    catch(IOException ex) {
        fileError(file, "file.open.error", ex);
    }
    catch (ClassNotFoundException ex) {
        fileError(file, "file.open.error", ex);
    }
    finally {
        if (ois != null) {
            try {
                ois.close();
            }
            catch (IOException ex) {
                Logger.getLogger(PersonView.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

メインウィンドウの取得

モデルを読み込んだものに置き替える(後述)

モデルの保存

PersonViewのsavePersonFile()を実装する。

```
    public void savePersonFile() {
        JFrame parent = getFrame();
        if (fileChooser.showSaveDialog(parent) != JFileChooser.APPROVE_OPTION) return;
        File file = fileChooser.getSelectedFile();
        ObjectOutputStream oos = null;
        try {
            oos = new ObjectOutputStream
                (new BufferedOutputStream(new FileOutputStream(file)));
            oos.writeObject(person);
        }
        catch (IOException ex) {
            fileError(file, "file.save.error", ex);
        }
        finally {
            if (oos != null) {
                try {
                    oos.close();
                }
                catch (IOException ex) {
                    fileError(file, "file.save.error", ex);
                }
            }
        }
    }
```

エラーメッセージの表示

```
void fileError(File file, String messageKey, Throwable cause) {
    JFrame parent = getFrame();
    ResourceMap resourceMap = Application.getInstance
        (PersonApp.class).getContext().getResourceMap(PersonView.class);

    JOptionPane.showMessageDialog
        (parent,
         String.format(resourceMap.getString(messageKey),
                       file.getAbsolutePath()));
    Logger.getLogger(PersonView.class.getName()).log(Level.SEVERE, null, cause);
}
```

メッセージプロパティの取得

PersonView.properties

```
file.save.error=Cannot write to file '%1$s'.
file.open.error=Cannot open file '%1$s'.

file.save.error=ファイルに書き込めません('%1$s')。
file.open.error=ファイルが開けません('%1$s')。
```

getString(): 文字列
getInteger(): int
getIcon(): Icon
...
injectFields(object) とすると、@Resource が付いたフィールドにインジェクトしてくれる。

モデルの更新

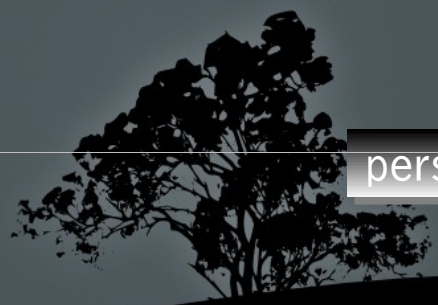
- コンポーネント経由で修正するか、
- あるいはバインドし直す

コンポーネント経由

```
    void storePerson(Person newPerson) {
        nameTextField.setText(newPerson.getName());
        ageTextField.setText(String.valueOf(newPerson.getAge()));
    }
```

バインドし直す

```
    void storePerson(Person newPerson) {
        for (Binding b: bindingGroup.getBindings()) {
            if (b.getSourceObject() == person) {
                b.unbind();
                b.setSourceObject(newPerson);
                b.bind();
            }
        }
        person = newPerson;
    }
```



person07

モデルの更新

- 「なんでモデルのプロパティの方を書き替えないの？」
  - 現在は単方向のバインド(ビュー ＝＞ モデル)しか行っていないため。
- つまり、プログラムでモデルを更新しても、ビューには自動的には反映されない。
  - 不便なので両方向のバインドにしましょう。

BeanInfo を作成する

- Person を作り直して、両方向のバインドが出来るようにする。

直接モデルを更新すればビューに反映されるようになる。

```
    void storePerson(Person newPerson) {
        nameTextField.setText(newPerson.getName());
        ageTextField.setText(String.valueOf(newPerson.getAge()));
    }
```

```
    void storePerson(Person newPerson) {
        try {
            BeanInfo bi = Introspector.getBeanInfo(person.getClass());
            for (PropertyDescriptor pd: bi.getPropertyDescriptors()) {
                if (pd.isBound()) {
                    Method reader = pd.getReadMethod();
                    Method writer = pd.getWriteMethod();
                    if (reader != null && writer != null) {
                        writer.invoke(person, reader.invoke(newPerson));
                    }
                }
            }
        }
        catch (IntrospectionException ex) {throw new RuntimeException(ex);}
        catch (IllegalAccessException ex) {throw new RuntimeException(ex);}
        catch (IllegalArgumentException ex) {throw new RuntimeException(ex);}
        catch (InvocationTargetException ex) {throw new RuntimeException(ex);}
    }
```

BeanInfo を利用して汎用的に記述しておくと、モデルへのプロパティ追加の時に書き直さなくても良くなる。

person08

- 07-08.ogg

入力チェック

- 2 つの観点がある
  - 型変換の問題（年齢 (int) に、"ABC" など）
    - バインディングエラーで捕捉
  - ビジネスルール（年齢が 1000 など）
    - バリデータで検証

まずは、エラーを表示するフィールドを作成しましょう。

- 08-09.ogg

入力チェック

- エラーチェックは、BindingListener を登録することで行う。

```
class BindingChecker implements BindingListener {
    public void bindingBecameBound(Binding binding) {}
    public void bindingBecameUnbound(Binding binding) {}

    public void syncFailed(Binding binding, SyncFailure failure) {
        ResourceMap resourceMap = Application.getInstance
            (PersonApp.class).getContext().getResourceMap(PersonView.class);
        if ("person.age".equals(binding.getName())) {
            if (failure.getType() == SyncFailureType.CONVERSION_FAILED) {
                ageErrorLabel.setText(resourceMap.getString("number.format.error");
            }
        }
    }

    public void synced(Binding binding) {
        if ("person.name".equals(binding.getName())) {
            nameErrorLabel.setText("");
        }
        else if ("person.age".equals(binding.getName())) {
            ageErrorLabel.setText("");
        }
    }

    public void sourceChanged(Binding binding, PropertyStateEvent event) {}
    public void targetChanged(Binding binding, PropertyStateEvent event) {}
}
```

バインドエラーの時

バインディング名(後述)

変換エラー

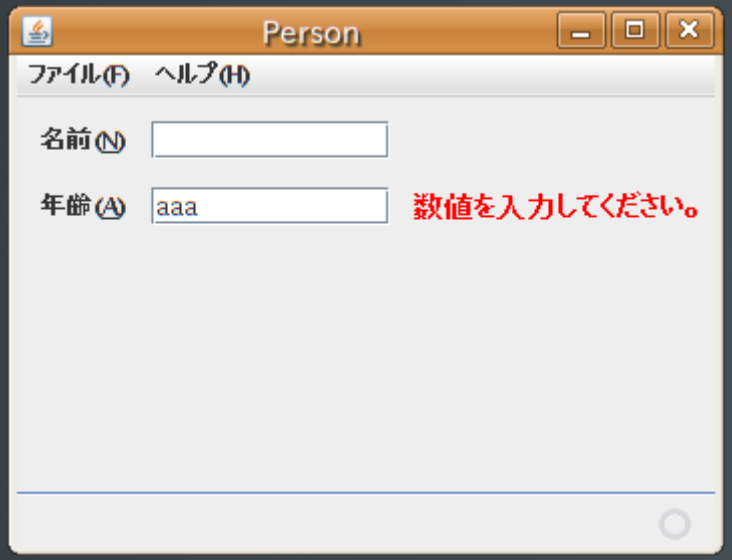
バインド成功の時

```
public PersonView(SingleFrameApplication app) {
    super(app);

    initComponents();
    bindingroup.addBindingListener(new BindingChecker());
```

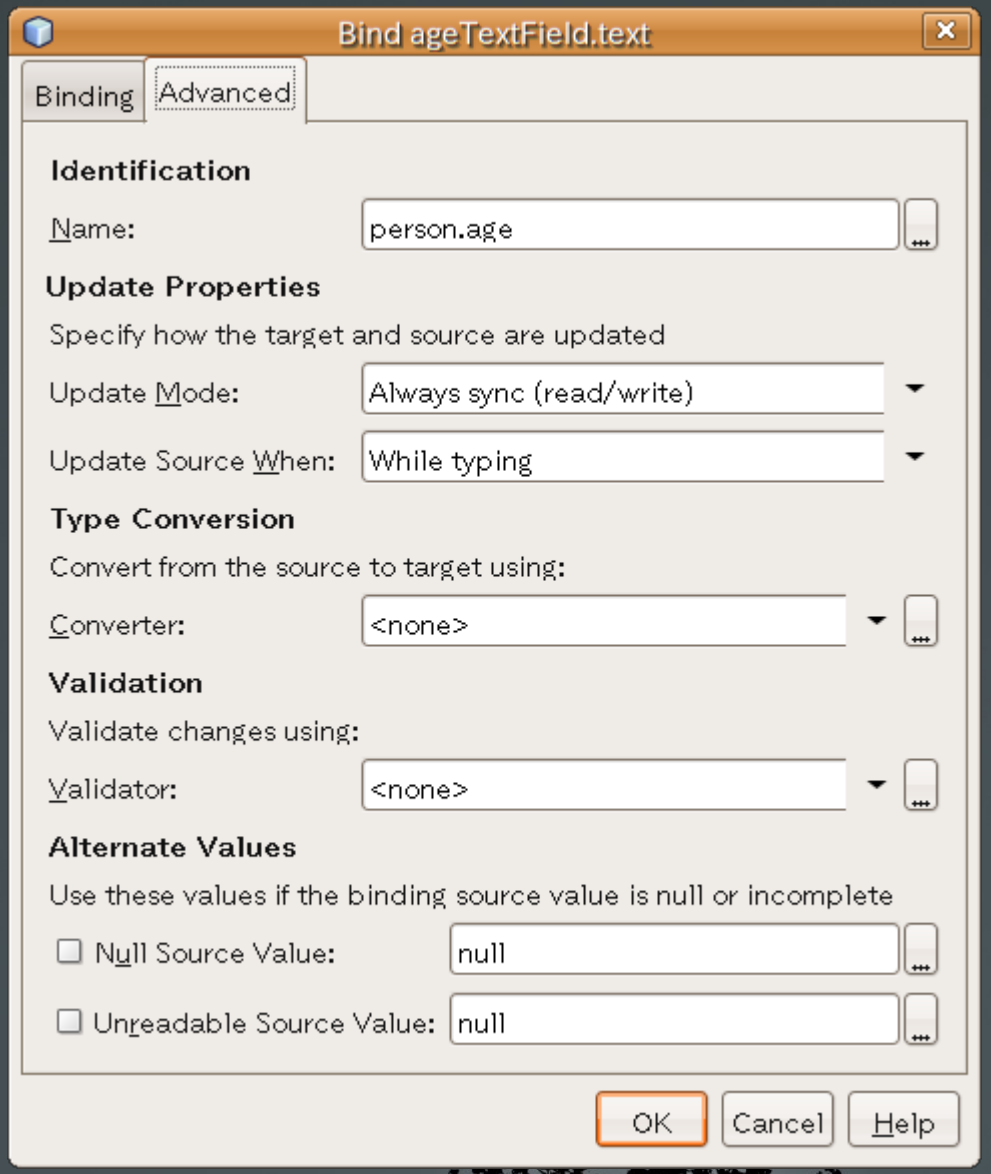
入力チェック

- バインディングに名前を付けて、
- エラーメッセージを用意。

```
number.format.error=Enter numeric value.

number.format.error=数値を入力してください。
```



person09



入力チェック

- バリデータを作成
  - MinMaxValidator
    値の範囲をチェック
  - RequiredValidator
    null でも長さ 0 でもないことをチェック

RequiredValidator

- 入力値が null でも長さ 0 でもないことをチェック
- Validator<T> を継承して作成する

```
public class RequiredValidator extends Validator<Object> {
    public enum ErrorCode {
        NULL_VALUE, LENGTH_ZERO
    }

    public final Validator.Result NULL_VALUE
        = new Validator.Result(ErrorCode.NULL_VALUE, "Null is not permitted.");
    public final Validator.Result LENGTH_ZERO
        = new Validator.Result(ErrorCode.LENGTH_ZERO, "Length zero.");

    @Override public Validator.Result validate(Object value) {
        if (value == null) return NULL_VALUE;
        if (value instanceof CharSequence &&
            ((CharSequence)value).length() == 0) return LENGTH_ZERO;
        return null;
    }
}
```

あらゆる型で利用可能

エラーを Validator.Result で定義

エラーコード

簡単な解説 (あまり意味は無い)

検証機能を実装

エラーが無ければ null を返す

バリデータの登録

- JavaBean として登録し、バインディングの設定の Advance に設定。
- BindingListener を変更。

```
if ("person.name".equals(binding.getName())) {
    if (failure.getType() == SyncFailureType.VALIDATION_FAILED) {
        nameErrorLabel.setText(resourceMap.getString("input.required.error"));
    }
}
else if ("person.age".equals(binding.getName())) {
    if (failure.getType() == SyncFailureType.CONVERSION_FAILED) {
        ageErrorLabel.setText(resourceMap.getString("number.format.error"));
    }
}
```

```
input.required.error=Input is required.

input.required.error=入力必須です。
```

- 09-10.ogg

バリデータの登録

- 初期段階ではバリデータが動作しないので、デフォルト値を設定しておく。

```
        initComponents();
        setInitialValue();
...

    void setInitialValue() {
        ResourceMap resourceMap = Application.getInstance
            (PersonApp.class).getContext().getResourceMap(PersonView.class);
        nameTextField.setText(resourceMap.getString("name.default"));
        ageTextField.setText(resourceMap.getString("age.default"));
    }
```

- 保存前にエラーチェック。

```
    public void savePersonFile() {
        JFrame parent = getFrame();
        for (Binding b: bindingGroup.getBindings()) {
            if (b.getTargetValueForSource().failed()) {
                ResourceMap resourceMap = Application.getInstance
                    (PersonApp.class).getContext().getResourceMap(PersonView.class);
                JOptionPane.showMessageDialog
                    (parent, resourceMap.getString("fix.error"));
                return;
            }
        }
...
```

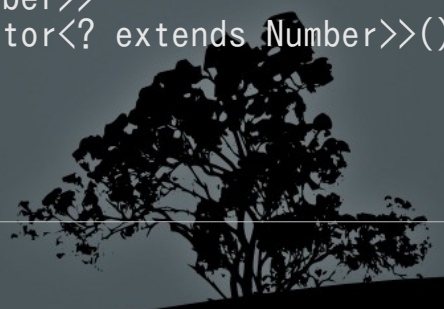
エラーの存在有無を確認

person10

MinMaxValidator

- 与えられた範囲内であることをチェック。

```
public class MinMaxValidator<T extends Number> extends Validator<T> {
    public enum ErrorCode {
        BELOW_MIN, ABOVE_MAX;
    }

    T min;
    T max;

    public void setMin(T min) {
        this.min = min;
    }

    public T getMin() {
        return min;
    }

    public void setMax(T max) {
        this.max = max;
    }

    public T getMax() {
        return max;
    }

    static final Map<Class<? extends Number>, Comparator<? extends Number>>
        comparatorTable = new HashMap<Class<? extends Number>, Comparator<? extends Number>>();
```



MinMaxValidator

```
    static {
        comparatorTable.put(Byte.class, new Comparator<Byte>()
            {public int compare(Byte b1, Byte b2) {return b1.compareTo(b2);}});
        comparatorTable.put(Integer.class, new Comparator<Integer>()
            {public int compare(Integer i1, Integer i2) {return i1.compareTo(i2);}});
        comparatorTable.put(Double.class, new Comparator<Double>()
            {public int compare(Double d1, Double d2) {return d1.compareTo(d2);}});
        comparatorTable.put(Float.class, new Comparator<Float>()
            {public int compare(Float f1, Float f2) {return f1.compareTo(f2);}});
        comparatorTable.put(Long.class, new Comparator<Long>()
            {public int compare(Long l1, Long l2) {return l1.compareTo(l2);}});
        comparatorTable.put(Short.class, new Comparator<Short>()
            {public int compare(Short s1, Short s2) {return s1.compareTo(s2);}});
    }

    public final Validator.Result BELOW_MIN
        = new Validator.Result(ErrorCode.BELOW_MIN, "Value is less than minimum value.");
    public final Validator.Result ABOVE_MAX
        = new Validator.Result(ErrorCode.ABOVE_MAX, "Value exceeds maximum value.");

    @Override
    public Validator<T>.Result validate(T value) {
        if (value == null) return null;
        Comparator<T> cmp
            = (Comparator<T>)comparatorTable.get(value.getClass());
        if (cmp == null)
            throw new RuntimeException("Unsupported type:" + value.getClass());
        if (cmp.compare(value, min) < 0) return BELOW_MIN;
        if (cmp.compare(max, value) < 0) return ABOVE_MAX;
        return null;
    }
}
```

MinMaxValidator の登録

- 同様に ageTextField のバインド設定に登録。
- 型付きバリデータなので、型パラメータを指定。
- 最大、最小をプロパティで指定。

- 10-11.ogg

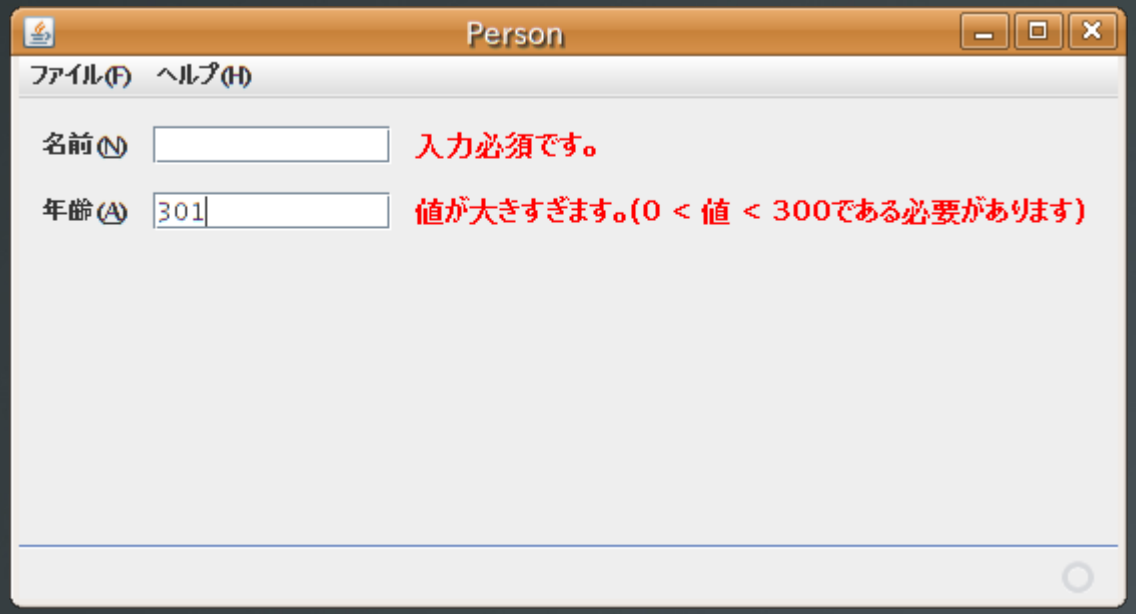
MinMaxValidator の登録

```
else if ("person.age".equals(binding.getName())) {
    if (failure.getType() == SyncFailureType.CONVERSION_FAILED) {
        ageErrorLabel.setText(resourceMap.getString("number.format.error"));
    }
    else if (failure.getType() == SyncFailureType.VALIDATION_FAILED) {
        if (failure.getValidationResult().getErrorCode() ==
            MinMaxValidator.ErrorCode.BELOW_MIN)
        {
            ageErrorLabel.setText
                (String.format(resourceMap.getString("min.value.error"),
                               ageValidator.getMin(),
                               ageValidator.getMax()));
        }
        else if (failure.getValidationResult().getErrorCode() ==
                 MinMaxValidator.ErrorCode.ABOVE_MAX)
        {
            ageErrorLabel.setText
                (String.format(resourceMap.getString("max.value.error"),
                               ageValidator.getMin(),
                               ageValidator.getMax()));
        }
        else {
            ageErrorLabel.setText(resourceMap.getString("invalid.value.error"));
        }
    }
}
```

BindingListener を変更。

min.value.error= 値が小さすぎます。(%1$,d < 値 < %2$,d である必要があります)
max.value.error= 値が大きすぎます。(%1$,d < 値 < %2$,d である必要があります)
invalid.value.error= 値が不適当です。

ひとまず完成。でも ...



一般の GUI アプリケーションとして見ると、ちょっと見劣りしますね。

person11

- 「開く」と「保存」しかない。
  - 「新規」、「開く」、「保存」、「名前を付けて保存」、「閉じる」が欲しい。
  - 値を保存しないで終了しても何も警告されない。
- Undo とかは？

SingleModelLifeCycleManager

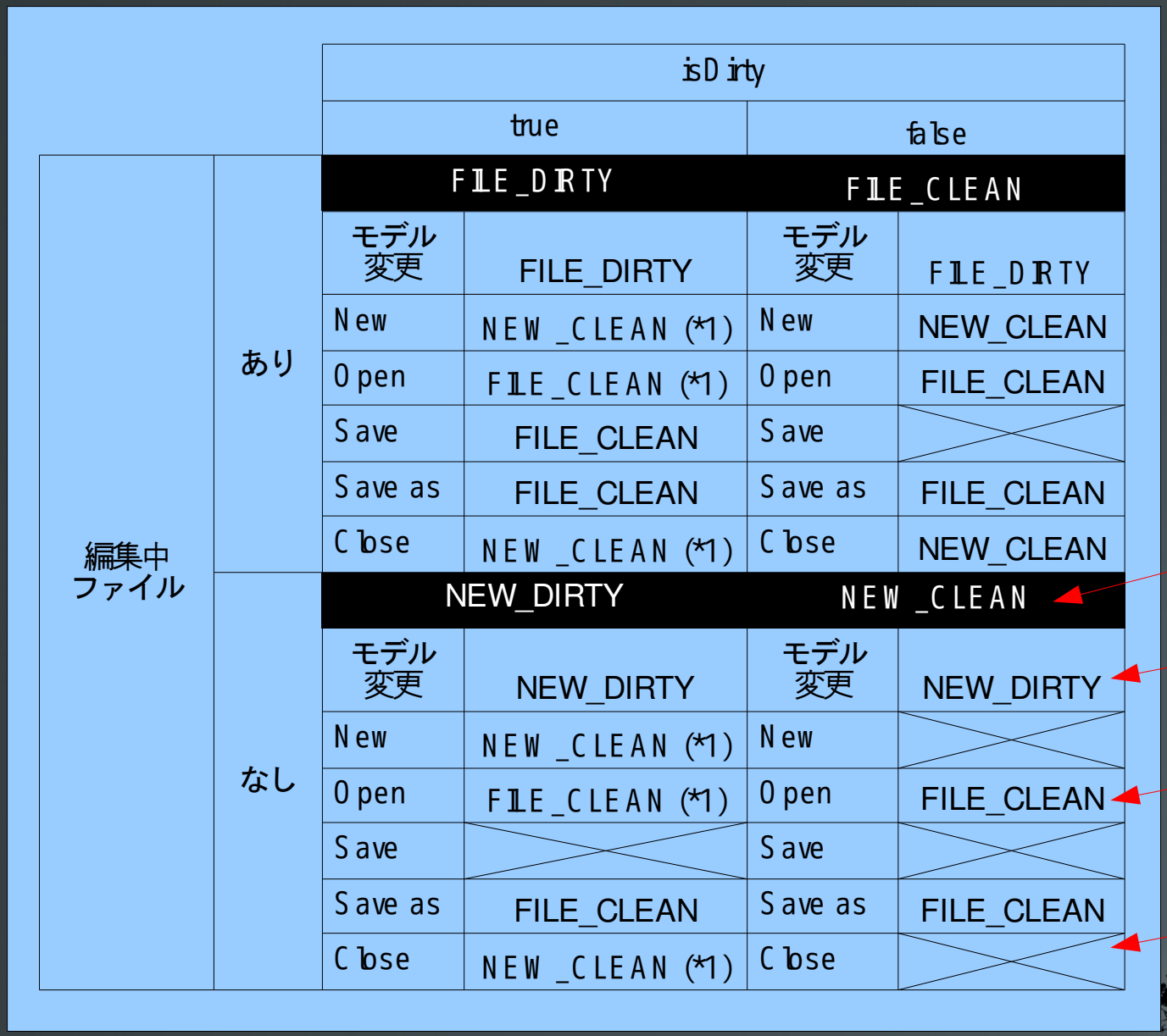
- 一般的な、アプリケーションの、モデルライフサイクル管理を JavaBeans にカプセル化してみる実験として、作成してみました。
- 提供機能
  - メニュー管理
  - モデルのライフサイクル管理
  - Undo 処理

制御系も JavaBeans を使って、バインド！

現在の NetBeans は、型パラメータ付き Bean の BeanInfo を編集できないので注意。(一旦 Object で作成、BeanInfo ができてから、<T> に置きかえ)

4 つの状態を定義

状態遷移図



(*1) 変更の保存をするかを問い合わせ。

SingleModelLifeCycleManager を使ってみる。

- ライブラリの登録、パレットへの登録 person12
  - 11-12.ogg
- メニューの追加 person13
  - 12-13.ogg
- バインド person14
  - 13-14.ogg (openPersonFil(), savePersonFile(), storePerson() を削除)
- イベントハンドラ
  - 14-15.ogg

person15

ModelLifeCycleContext

ModelLifeCycleManager の求めに応じて、コンテキストを提供する。

YesNoCancelResponse querySaveCurrentModel()
　現在の変更を保存するか問い合わせ。

YesNoCancelResponse queryOverwrite(File file)
　ファイルを上書きするか問い合わせ。

YesNoCancelResponse queryOverwrite(File file)
　ファイルを上書きするか問い合わせ。

File querySaveFile()　ファイルの保存先を問い合わせ。

File queryOpenFile()　開くファイルを問い合わせ。

T loadModel(File file)　モデルをファイルから読み出し。

boolean saveModel(File file, T model)　モデルをファイルに書き込み。

void setInitialValue(T model)　モデルを新規生成した時に初期値を設定。

boolean isModelValid(BindingGroup bindingGroup)
　現在編集中のモデルにエラーが無いかをチェック。

SimpleModelLifeCycleContext

デフォルト実装を提供する。モデルは Java の直列化で保存される。
必要に応じて継承して実装を変更可能。

```
    class PersonModelLifeCycleContext extends SimpleModelLifeCycleContext<Person> {
...
        @Override public void setInitialValue(Person person) {
            ResourceMap resourceMap = Application.getInstance
                (PersonApp.class).getContext().getResourceMap(PersonView.class);
            person.setName(resourceMap.getString("name.default"));
            person.setAge(resourceMap.getInteger("age.default"));
        }
```

モデルの値を初期化するコード
setInitialValue() と同様

```
PersonModelLifeCycleContext
(Component parent,
 String saveQueryTitle, String saveQueryMessage, Icon saveQueryIcon,
 String queryOverwriteTitle, String queryOverwriteMessage, Icon queryOverwriteIcon,
 String fixErrorTitle, String fixErrorMessage, Icon fixErrorIcon,
 String fileSaveErrorTitle, String fileSaveErrorMessage, Icon fileSaveErrorIcon,
 String fileOpenErrorTitle, String fileOpenErrorMessage, Icon fileOpenErrorIcon,
 JFileChooser querySaveFileChooser,
 JFileChooser queryOpenFileChooser)
```

ダイアログの親

保存確認

上書き確認

エラー通知

保存エラー

オープンエラー

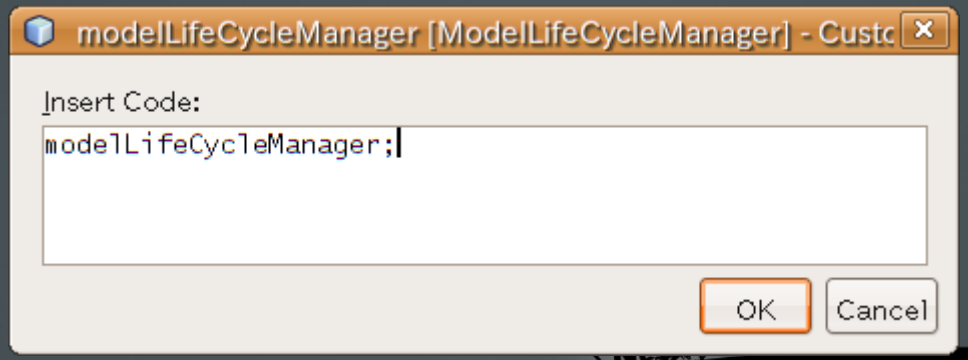
保存用 FileChooser

オープン用 FileChooser

ModelLifeCycleContext 生成と登録

```
        modelLifeCycleManager = new ModelLifeCycleManager<Person>();
        ResourceMap resourceMap = getResourceMap();
        JFileChooser saveFileChooser = new JFileChooser();
        saveFileChooser.setMultiSelectionEnabled(false);

        JFileChooser openFileChooser = new JFileChooser();
        openFileChooser.setMultiSelectionEnabled(false);

        PersonModelLifeCycleContext context = new PersonModelLifeCycleContext
            (getFrame(), resourceMap.getString("save.query.title"),
             resourceMap.getString("save.query.message"),
             resourceMap.getIcon("save.query.icon"),
...
             saveFileChooser, openFileChooser);
        modelLifeCycleManager.start(Person.class, context);

        initComponents();
```

modelLifeCycleManager が initComponents() の中で上書きされないように、modelLifeCycleManager の Custom Creation Code を指定。



アプリケーション終了時の処理

```
initComponents();

getApplication().addExitListener(new ExitListener() {
    public boolean canExit(EventObject event) {
        return modelLifeCycleManager.invokeCloseModel();
    }

    public void willExit(EventObject event) {}
});
```

addExitListener() で、終了時に行う処理を追加できる。

canExit() で false を返せば、終了をやめることができる。

person16

ツールバーの追加

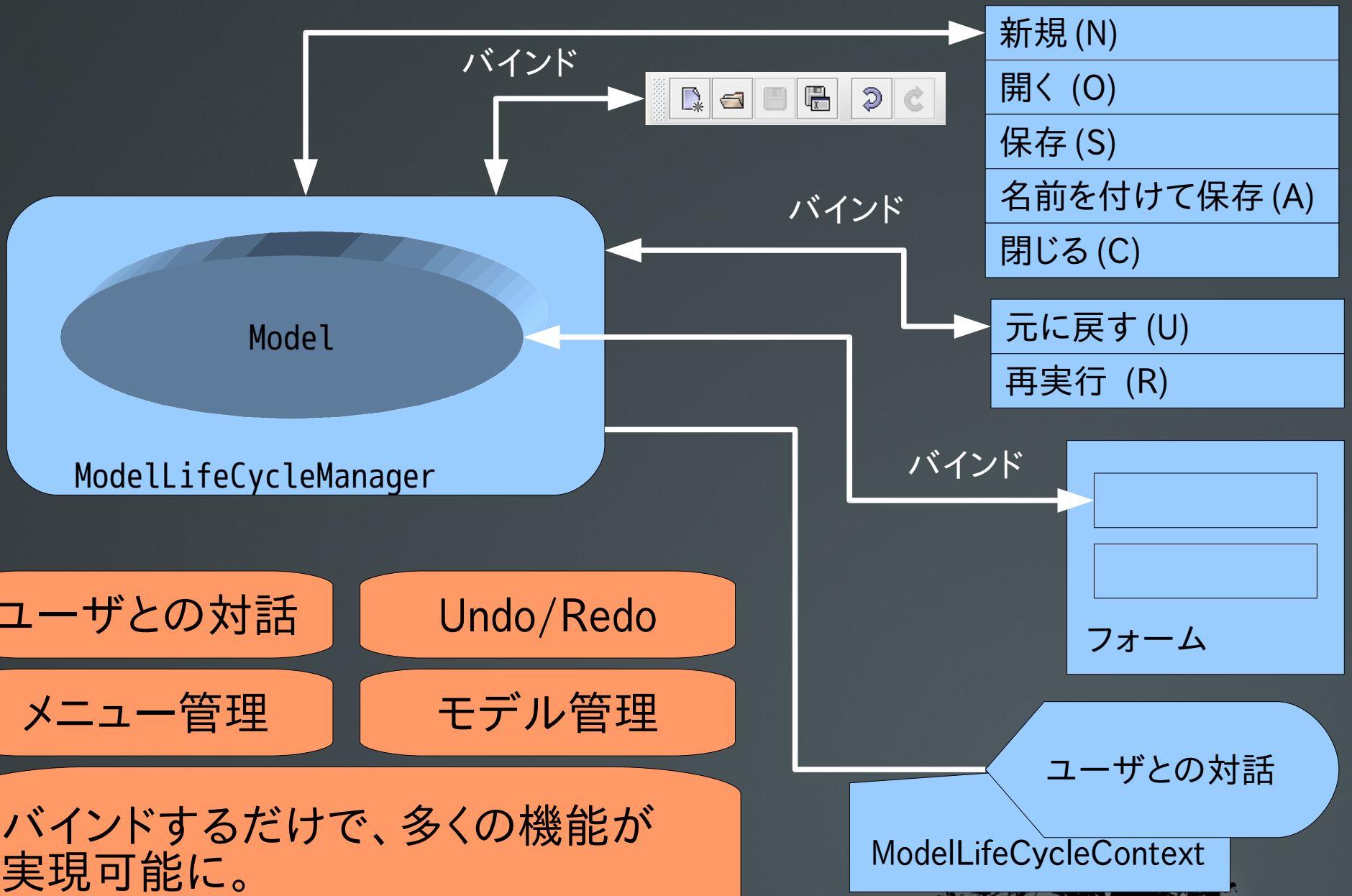
- [Frame View] のプロパティで追加できる。
- アイコンは "Java look and feel Graphics Repository" を利用。
- jlfgr-1_0.jar をクラスパスに追加しておく。
- リソースファイルでクラスパス内の場所を指定。

person17

```
newPersonButton.icon=/toolbarButtonGraphics/general/New24.gif
openPersonButton.icon=/toolbarButtonGraphics/general/Open24.gif
saveButton.icon=/toolbarButtonGraphics/general/Save24.gif
saveAsButton.icon=/toolbarButtonGraphics/general/SaveAs24.gif
undoButton.icon=/toolbarButtonGraphics/general/Undo24.gif
redoButton.icon=/toolbarButtonGraphics/general/Redo24.gif
```

- 16-17.ogg

ModelLifeCycleManager に見るバインディングの可能性



ユーザとの対話

Undo/Redo

メニュー管理

モデル管理

バインドするだけで、多くの機能が実現可能に。

まとめ

- Swing Application Framework と IDE で、リソースの管理や、アクションの記述が大幅に簡単になる。
- Beans Binding で、フォームベースのアプリケーションが簡単に。
- 制御系のコンポーネントも、Java Beans で提供することで、バインドによってアプリケーションが構築可能に。
- ただ、裏で何が起きているか分かりにくいので、ソースを片手にハック！

Java によるリッチクライアント

- 遅い
  - 今の Swing は速いし、JVM 起動も、ネットワークも速くなった！
- 開発が面倒
  - IDE と JSR295/JSR296 で大幅に楽に！
- デプロイが面倒
  - Java Web Start で簡単。マルチプラットフォーム！
- でも JRE のデプロイが
  - Java SE 6 Update N で改善！
- ロジック記述やデータアクセスが面倒
  - JVM 上で動く別言語という選択肢 (C
  - JPA

今一度、見直してみては？

謝辞

- ねこび～ん
  http://ja.netbeans.org/nekobean/

- Java look and feel Graphics Repository
  http://java.sun.com/developer/techDocs/hi/repository/index.html
  標準的なアイコン集