2012 CEE-SEC(R)
Software Engineering Conference in Russia

CUSTIS®

Трансформируем автоформы в качественный интерфейс с помощью DSL

Иван Гаммель (igammel@custis.ru),
Ведущий разработчик

Москва, ноябрь 2012

CUSTIS®

Коротко о главном

- Понятия «Быстро» и «качественно» не исключают друг друга, когда речь идет о формах
- Совмещение двух подходов – генерация + DSL – дает интересный результат
- Это может сделать каждый так, как это сделали мы

CUSTIS®

Чего нет в этом докладе?

- Рассказа о юзабилити форм
- Приемов построения предметно-ориентированных языков (DSL)
- Алгоритмов генерации кода
- Java

CUSTIS®

Тогда о чем он?

- О постановке задачи
- Об идее связывания генерации интерфейса и DSL
- Об архитектуре возможного решения
- О качестве полученного кода

- *Примеры в этом докладе содержат код, использующий уже существующее решение – библиотеку CUSTIS Rich Forms.

CUSTIS®

Нужен способ, который…

- Избавляет от рутины и копипасты
- (лишний код, лишние ошибки),
- Тем самым уменьшая объем работы
- И повышая качество продукта
- …
- PROFIT

CUSTIS®

Получается постановка задачи, если добавить:

- Необходимость быстро реагировать на изменения
- Возможность использования широким кругом разработчиков с различной квалификацией и опытом
- Качественный пользовательский интерфейс на выходе

CUSTIS®

Есть разные подходы

- Пишем формы вручную, используя компоненты из внешних библиотек
- Используем визуальный редактор, автоматически генерируем исходный код

CUSTIS®

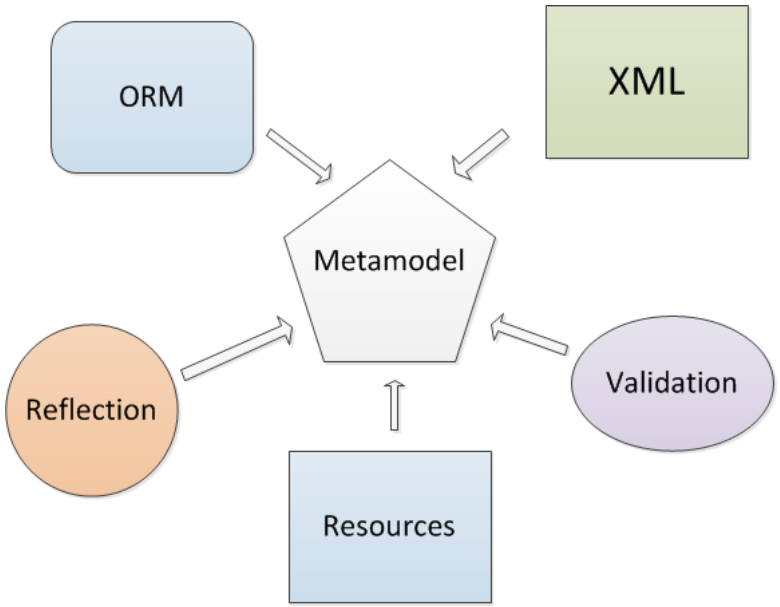
Плюсы/минусы?

- Вручную дорого и длинно, частое дублирование кода.
  - Но: максимальная свобода действий.
- Визуальный редактор/генерация кода: не гибко, трудно работать с кодом, не всегда возможен round-trip, новые зависимости.
  - Но: очень высокая скорость решения типовых задач.

CUSTIS®

Наблюдения

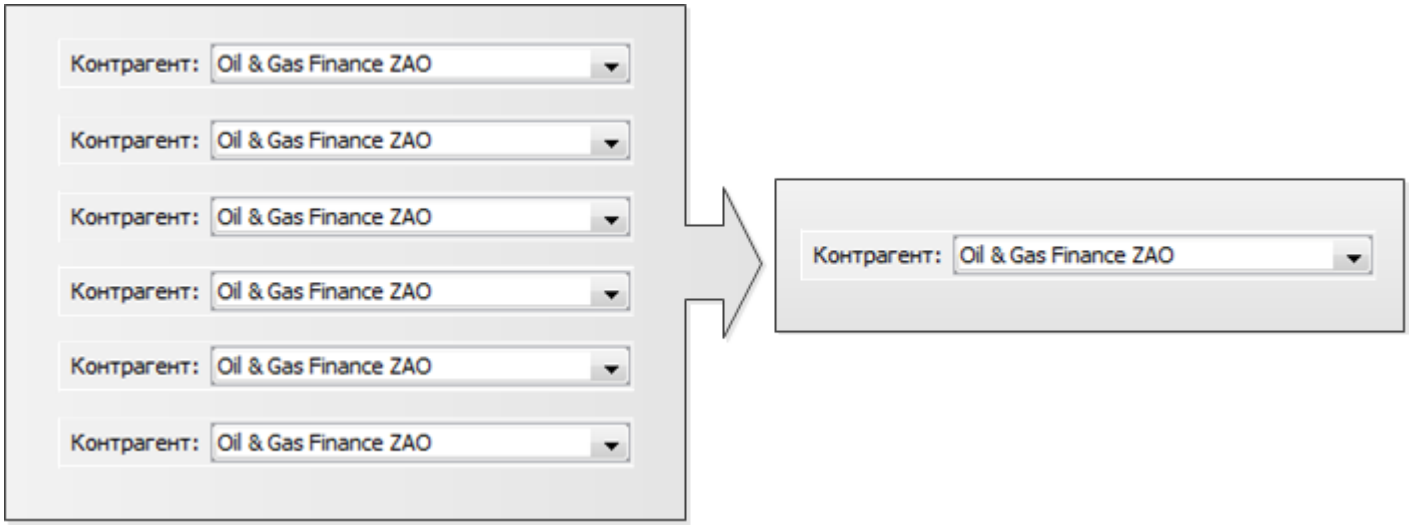
- В проекте много метаданных, которые можно повторно использовать (например, в генераторе форм)



CUSTIS®

Наблюдения

- В пределах одного проекта формы похожи друг на друга – сходные верстка, дизайн и механизмы обратной связи



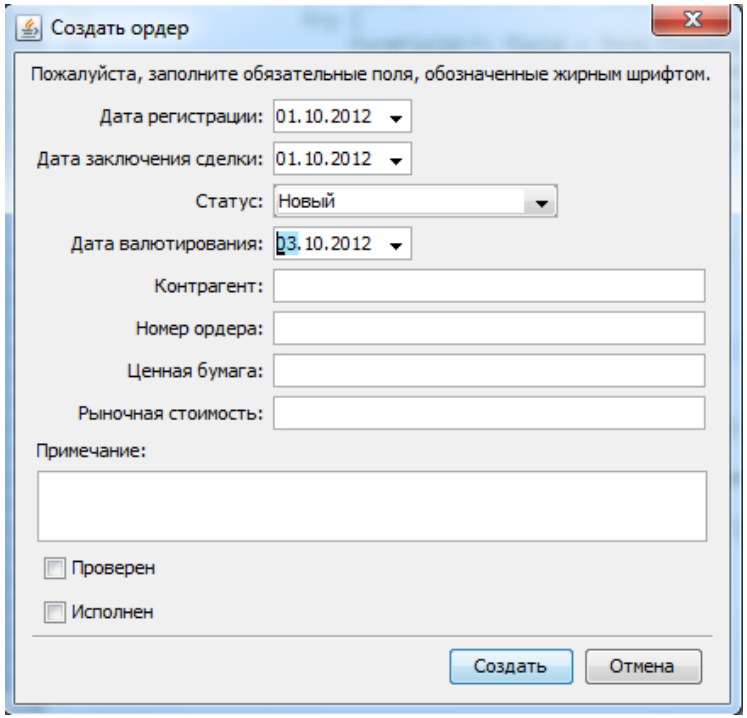
CUSTIS®

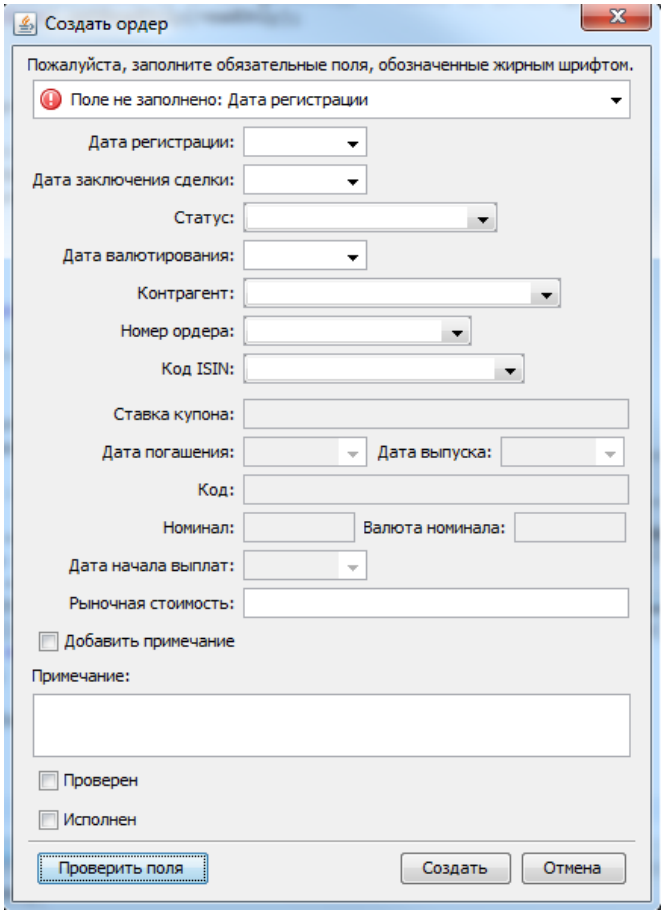
Идея! А что если совместить подходы?

- Быстро выполняем задачу с помощью генерации кода
- Получаем дополнительную свободу действий за счет DSL и расширяемости

CUSTIS®

Как это может выглядеть: до и после





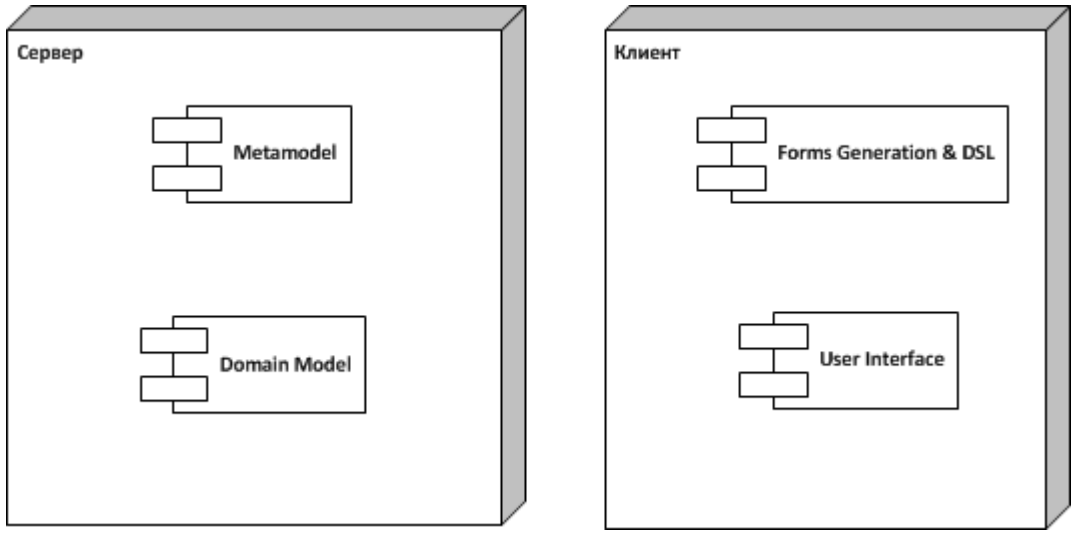
CUSTIS®

Подходы к архитектуре

- **DDD** и **TDD** в пользовательском коде
- **DSL** – повышаем выразительность и читабельность
- Модульность и расширяемость – не пытаемся отвечать на вопросы о жизни, вселенной и вообще.
- Вместо монолитного фреймворка две маленьких библиотеки: метамодель и генератор форм

CUSTIS®

Пример компонентной архитектуры приложения



CUSTIS®

Метамодель: поиск решения

- OMG MOF, EMF – сложно
- Java Reflection – недостаточно
- Нужно:
  - иерархия классов,
  - свойства,
  - множественность связи,
  - reference/containment,
  - использование пользовательских метаданных.

CUSTIS®

Метамодель: DIY

- Базовые сущности
  - Класс
  - Свойство
  - Примитив*
  - Перечисление
- Отношения
  - Наследование
  - Агрегирование/ссылка

CUSTIS®

Метамодель: расширяемость

- Интерфейс Mixed (примеси в Java)

```
Extension ext = new Extension();

IExtension value = model.extend(IExtension.class, ext);

assertSame(ext, value);

assertSame(value, model.as(IExtension.class));

assertTrue(model.hasFeaturesOf(IExtension.class));
```

- Добавляем в метамодель произвольные метаданные

CUSTIS®

Строим форму быстро и легко

- Использование DSL и генератора уменьшает объем кода в несколько раз
- Просто переводим постановку на английский и расставляем пунктуацию
- Пример:
  - Поле «Ценная бумага» использует список бумаг, отображаемых в виде кода ISIN.
  - Form field “Security” uses list of securities displayed as ISIN code.
  - В коде:

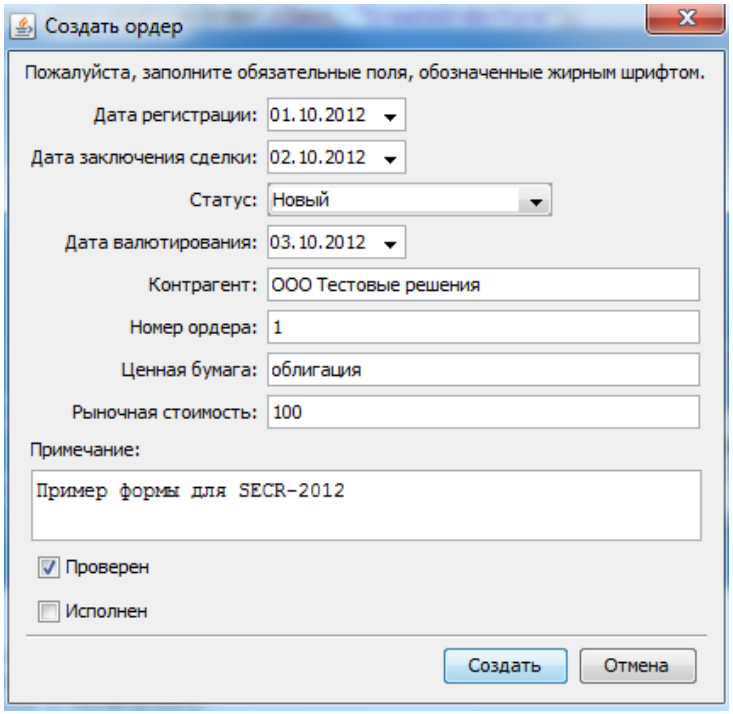
```
form.field("security", Security.class).useDataSource(securities).displayAs("isinCode")
```

CUSTIS®

Строим форму за 5 шагов с помощью DSL

- Шаг 1: Сгенерировать на основе метамодели как-нибудь, чтобы работало

```
final MDForm<Order> form = formFactory.generateForm(Order.class, "CreateOrderForm");
```



CUSTIS®

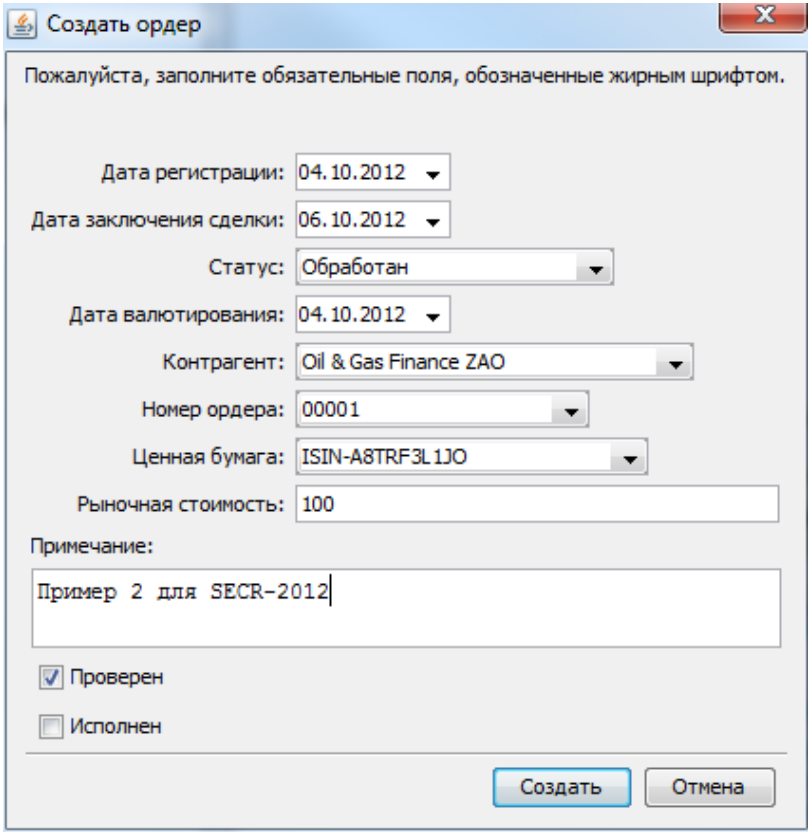
Строим форму за 5 шагов с помощью DSL

- Шаг 2: Задать источники данных

```
form.field("counterparty", Counterparty.class)
    .useDataSource(counterparties);

form.field("security", Security.class)
    .useDataSource(securities)
    .displayAs("isinCode");

form.field("number", String.class)
    .useDataSource(list("1", "2", "3", "4"))
    .renderUsing(numberFormatter);
```



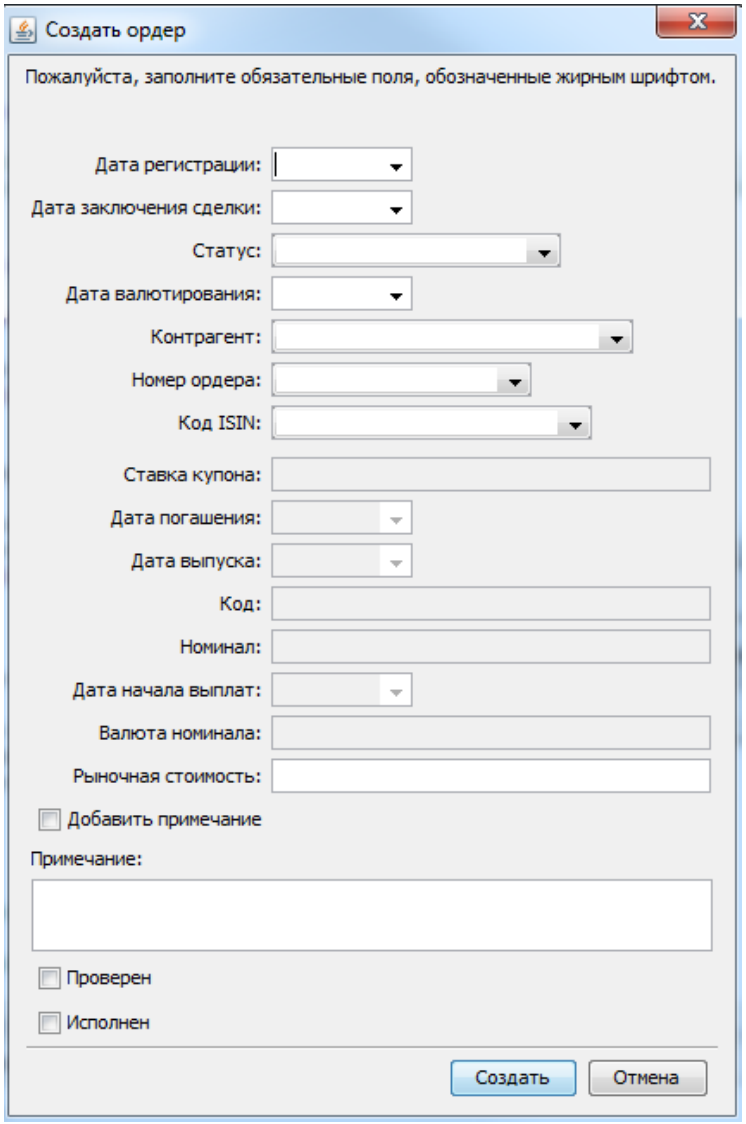
CUSTIS®

Строим форму за 5 шагов с помощью DSL

- Шаг 3: Добавить необходимые поля или убрать лишние

```
MDFieldGroup<Security> securityDetails = formFactory
            .generateFormGroup(Security.class, "securityDetails")
            .asReadOnly()
            .remove("isinCode")
            .bindTo("security");

form.addAfter(securityDetails, "security");
```

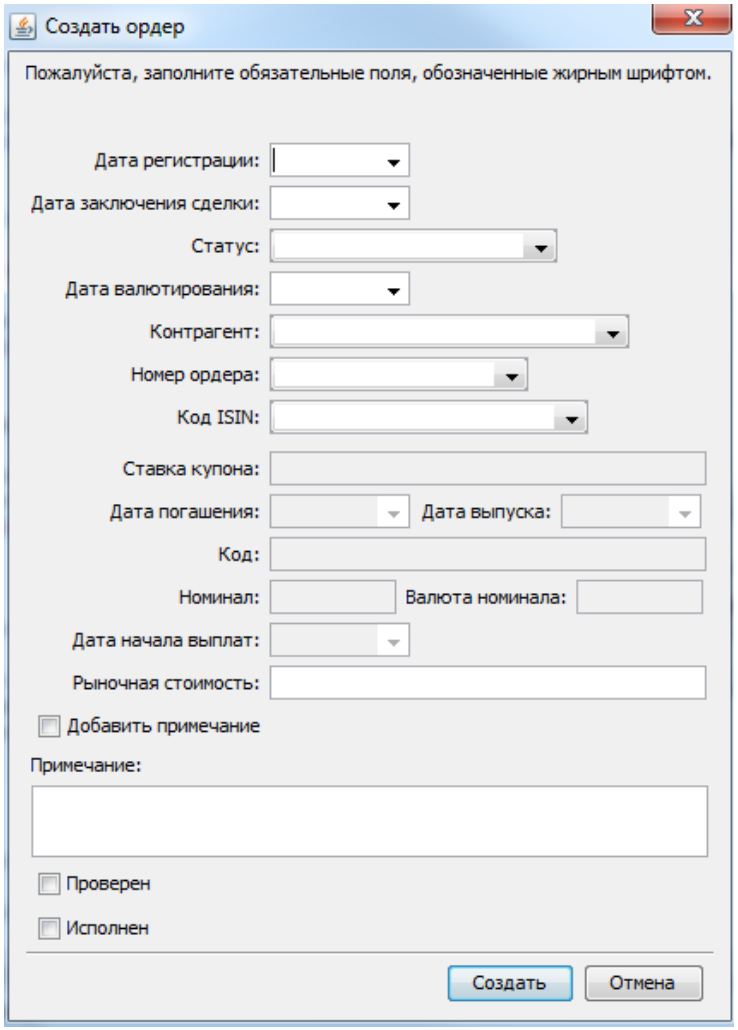


CUSTIS®

Строим форму за 5 шагов с помощью DSL

- Шаг 4: Настроить внешний вид

```
securityDetails
    .wrap("Dates","issueDate", "maturityDate")
    .decorate(layout(HORIZONTAL));
```



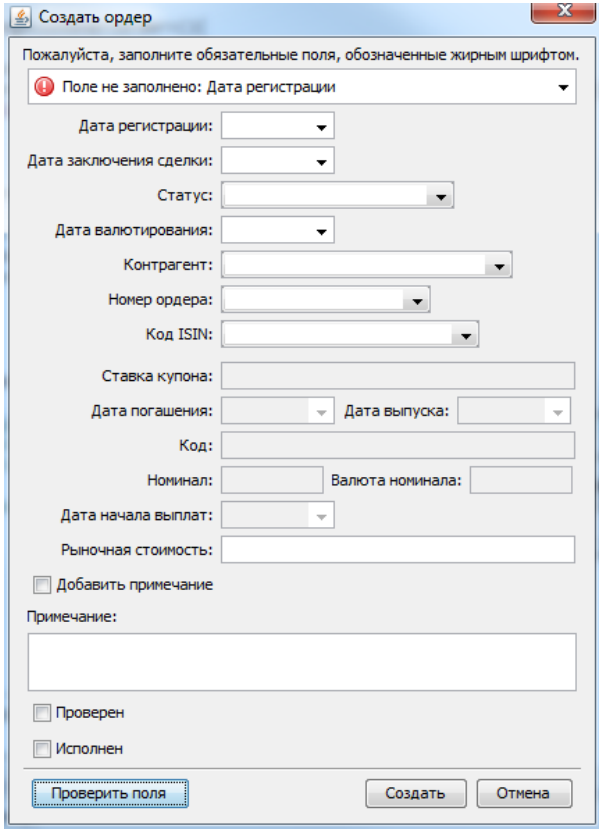
CUSTIS®

Строим форму за 5 шагов с помощью DSL

- Шаг 5: Реализовать проверку корректности ввода

```
form.add(action("validate", 10, orderValidator()));

form.field("number",String.class)
    .addChangeListener(notNullValidator());
```



CUSTIS®

Строим форму за 5 шагов с помощью DSL

- Сгенерировать на основе метамодели как-нибудь, чтобы работало
- Задать источники данных
- Добавить необходимые поля или убрать лишние
- Реализовать проверку корректности ввода
- Настроить внешний вид, если необходимо

CUSTIS®

Строим форму: TDD

- Интерфейс тестировать сложно даже с фреймворками.
- Можно тестировать логику формы, в том числе связанные поля, проверку корректности, нажатия кнопок.
- Для тестов необходима хорошая декомпозиция и соблюдение SRP, поэтому используем MVC

CUSTIS®

Строим форму: TDD

- Контроллер по модели построит и настроит представление, но это уже забота библиотеки.

- Пример:
  - При выборе ценной бумаги в поле «Дата выпуска» отображается дата выпуска ценной бумаги.

```
form.field("security", Security.class).setValue(selectedSecurity);
Calendar issueDate = form.field("issueDate", Calendar.class).getValue();
assertEquals(selectedSecurity.getIssueDate(), issueDate);
```

CUSTIS®

А как же качественный интерфейс?

- Проблемы:
  - Программисты не умеют «готовить» UX
  - У пользователей есть привычки, иногда – к ужасному*
  - Бюджет проекта – суровая реальность с эшелонированной обороной

CUSTIS®

Строим интерфейс

- Разумные настройки по умолчанию: следуем гайдлайну ОС*
- Консистентность: формы строятся по общим правилам, пользователь может не ждать сюрпризов.
- Простота и гибкость: если сгенерированная форма не удобна, её можно «подкрутить»: задать группировку и порядок полей, определить стили.

CUSTIS®

UX и код

- Разработчик прикладного ПО –пользователь библиотеки
- Решения проблем известны, но о них нужно помнить
  - Как подключить библиотеку? – Maven
  - Как использовать в контейнере DI и без него? – «нет» синглтонам, «да» фабрикам и разумным умолчаниям
  - Как тестировать код? – SRP, интерфейсы, модульность, поддержка JUnit и TestNG
  - Как найти нужный функционал? – naming conventions, поддержка autocomplete в IDE,

CUSTIS®

Заключение

- Все слагаемые важны для успеха:

Метамодель + автоформы + DSL = ♥

- Можно реализовать самостоятельно – это несложно и быстро.
- Библиотека CUSTIS Rich Forms используется в реальных проектах по созданию банковского ПО с мая 2012 года

CUSTIS®

Спасибо!

Вопросы?

Иван Гаммель
igammel@custis.ru
ivan-gammel.moikrug.ru

CUSTIS®

Ссылки по теме

- OMG MOF (MetaObject Facility)

www.omg.org/mof

- Eclipse Modeling Framework

www.eclipse.org/modeling

- «Предметно-ориентированные языки программирования», Мартин Фаулер

http://www.ozon.ru/context/detail/id/6967089/

- Windows 7/Vista UX Guidelines

http://www.microsoft.com/en-us/download/details.aspx?id=2695