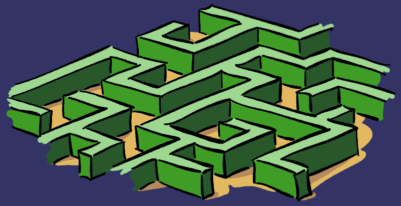
JAOO, Brisbane & Sydney, 2008

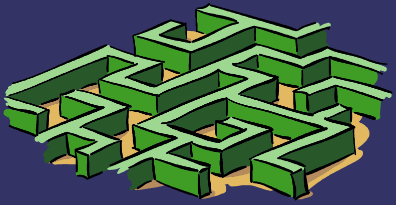
Componentisation in the Web Presentation Layer

Philip Lopez
Suncorp



Overview

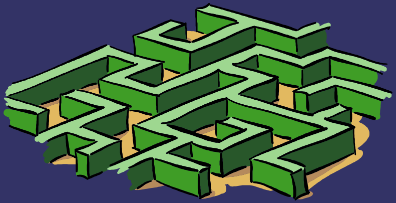
- Some problems facing ‘large’ organisations
  - A viewpoint on the **Suncorp** experience
  - Can web ‘component’ approaches help?
- A few code examples
  - SpringMVC 2.5.x
  - Tapestry 5 (beta)
  - Wicket 1.3.x
- Web components, SOA, and usability
- Future directions?



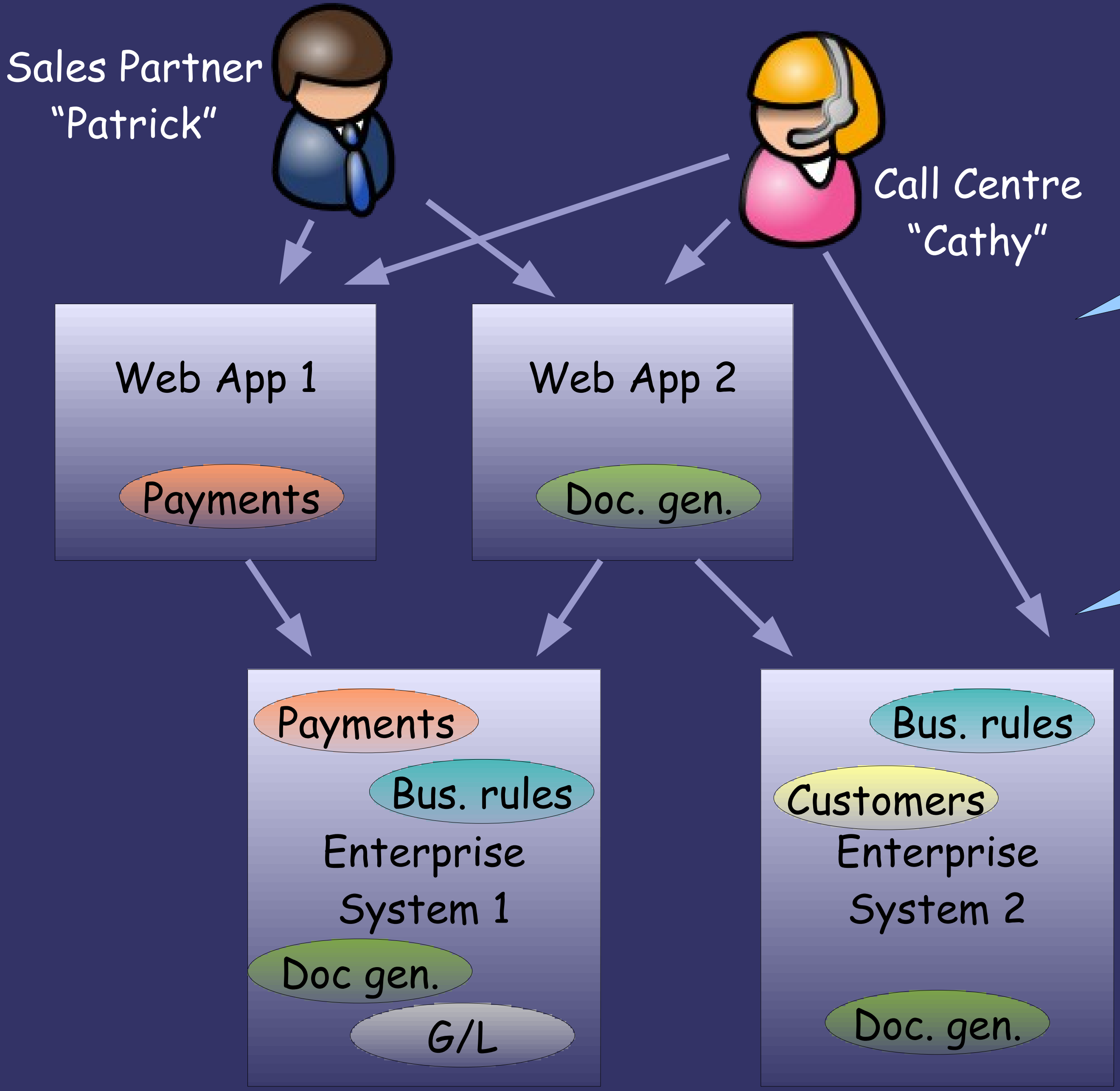
Many large organisations have a diverse (web) application portfolio

"It's better to build one thing many times than many things once"

- Business product or process centric
  - Not *user*-centric - switch between many apps
    - long training periods, frustration – labour market
- Expensive to maintain
  - Each system has **minimal** feature set
- Significant functional overlap
  - Different channels have similar needs

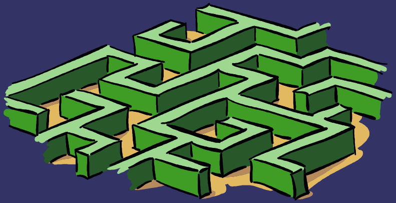


‘Typical’ application portfolio



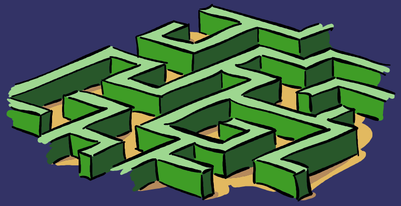
This is, of course, an **oversimplification**

Probably not a “win” for anyone

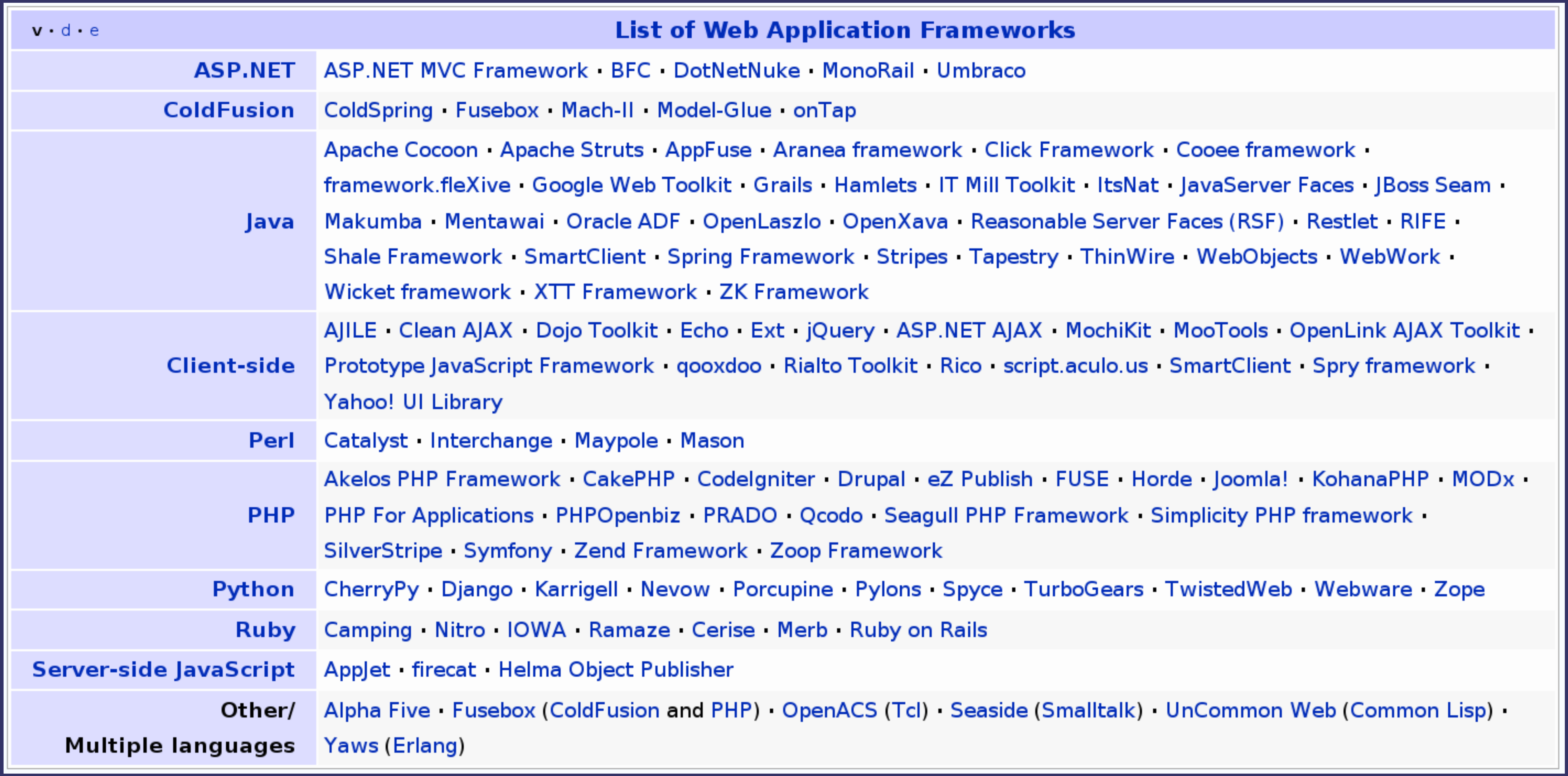


Obvious need for reuse, and yet...

- Reuse is still not commonplace
- Building for reuse takes more time/effort
- Product management approach required
- But some great examples:
  - Open source libraries
  - The RESTful web, mash ups, etc.
  - Maven 2
    - An example of the masochism we're willing to endure! :-)

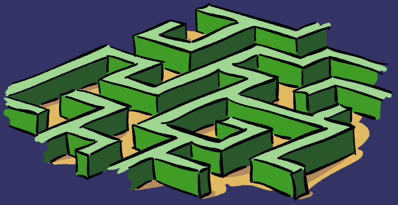


Only a few approaches to consider...



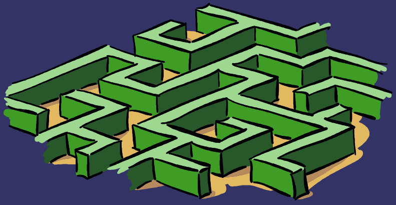
v · d · e	List of Web Application Frameworks
ASP.NET	ASP.NET MVC Framework · BFC · DotNetNuke · MonoRail · Umbraco
ColdFusion	ColdSpring · Fusebox · Mach-II · Model-Glue · onTap
Java	Apache Cocoon · Apache Struts · AppFuse · Aranea framework · Click Framework · Cooee framework · framework.fleXive · Google Web Toolkit · Grails · Hamlets · IT Mill Toolkit · ItsNat · JavaServer Faces · JBoss Seam · Makumba · Mentawai · Oracle ADF · OpenLaszlo · OpenXava · Reasonable Server Faces (RSF) · Restlet · RIFE · Shale Framework · SmartClient · Spring Framework · Stripes · Tapestry · ThinWire · WebObjects · WebWork · Wicket framework · XTT Framework · ZK Framework
Client-side	AJILE · Clean AJAX · Dojo Toolkit · Echo · Ext · jQuery · ASP.NET AJAX · MochiKit · MooTools · OpenLink AJAX Toolkit · Prototype JavaScript Framework · qooxdoo · Rialto Toolkit · Rico · script.aculo.us · SmartClient · Spry framework · Yahoo! UI Library
Perl	Catalyst · Interchange · Maypole · Mason
PHP	Akelos PHP Framework · CakePHP · CodeIgniter · Drupal · eZ Publish · FUSE · Horde · Joomla! · KohanaPHP · MODx · PHP For Applications · PHPOpenbiz · PRADO · Qcodo · Seagull PHP Framework · Simplicity PHP framework · SilverStripe · Symfony · Zend Framework · Zoop Framework
Python	CherryPy · Django · Karrigell · Nevow · Porcupine · Pylons · Spyce · TurboGears · TwistedWeb · Webware · Zope
Ruby	Camping · Nitro · IOWA · Ramaze · Cerise · Merb · Ruby on Rails
Server-side JavaScript	AppJet · firecat · Helma Object Publisher
Other/ Multiple languages	Alpha Five · Fusebox (ColdFusion and PHP) · OpenACS (Tcl) · Seaside (Smalltalk) · UnCommon Web (Common Lisp) · Yaws (Erlang)

Source: Wikipedia, accessed 27 May, 2008

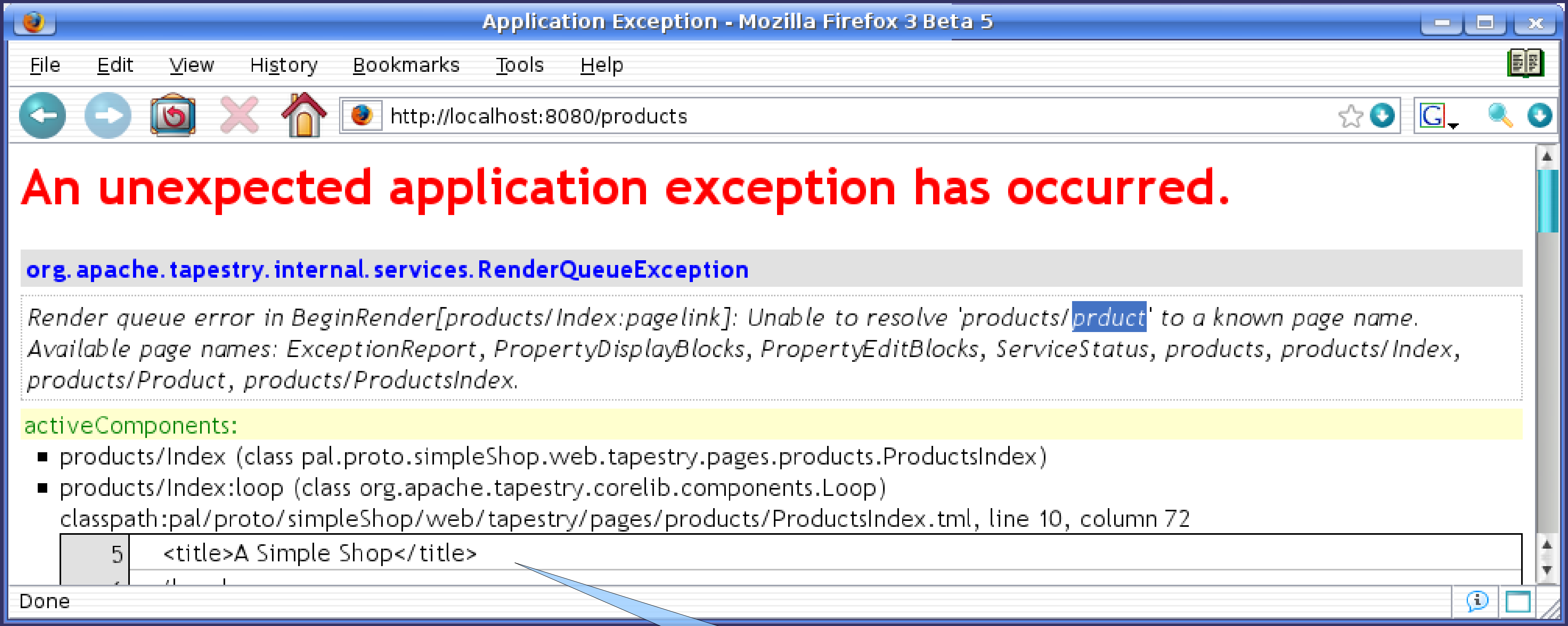


What do ‘web components’ offer?

- Reuse across applications / channels
  - Still reuse at other layers (e.g., SOA)
- Rapid development / assembly
  - But using high quality pre-built components
- Consistent user interfaces
  - ‘Standards’... <u>codified</u>, no longer *shelfware*
- Smaller units to understand, develop & test
- Higher levels of abstraction
  - Improved error detection
  - Improved code durability??



Example of error reporting


Application Exception - Mozilla Firefox 3 Beta 5
File Edit View History Bookmarks Tools Help
http://localhost:8080/products
Done

An unexpected application exception has occurred.

org.apache.tapestry.internal.services.RenderQueueException

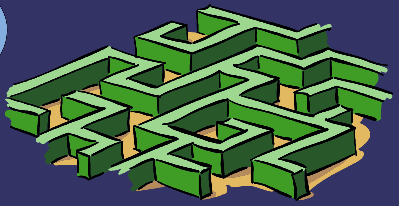
Render queue error in BeginRender[products/Index:pagelink]: Unable to resolve 'products/prduct' to a known page name. Available page names: ExceptionReport, PropertyDisplayBlocks, PropertyEditBlocks, ServiceStatus, products, products/Index, products/Product, products/ProductsIndex.

activeComponents:

- products/Index (class pal.proto.simpleShop.web.tapestry.pages.products.ProductsIndex)
- products/Index:loop (class org.apache.tapestry.corelib.components.Loop)
  classpath:pal/proto/simpleShop/web/tapestry/pages/products/ProductsIndex.tml, line 10, column 72

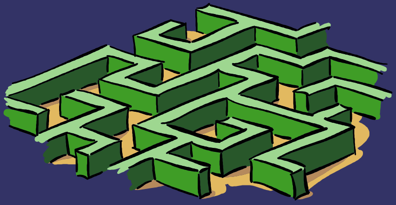
```
5   <title>A Simple Shop</title>
```

Error occurs on ‘calling’ page



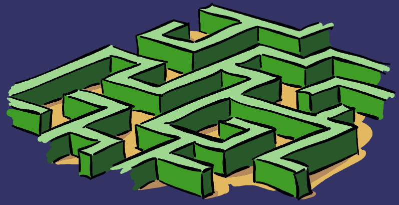
A quick review of HTTP

- GET – “idempotent” request
  - Ideal for “<u>render</u>” current resource/app **state**
  - URLs appropriate for bookmarks (context-rich)
- POST – not idempotent
  - Ideal for “<u>actions</u>” as they intend to change state



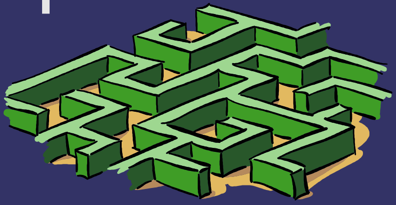
Render requests – the HTML ‘page space’

- **Render** requests output a HTML tree
  - ‘Stream-oriented’ approaches: “single-pass”
  - DOM-oriented – random access
- “Contributing” to the DHTML page space
  - ‘extension points’ such as HEAD, end of BODY
- Namespace uniqueness concerns
- Sub-spaces (distinct namespaces)
  - CSS
  - JavaScript
- DOM can be modified at runtime
  - Basis for DHTML/AJAX

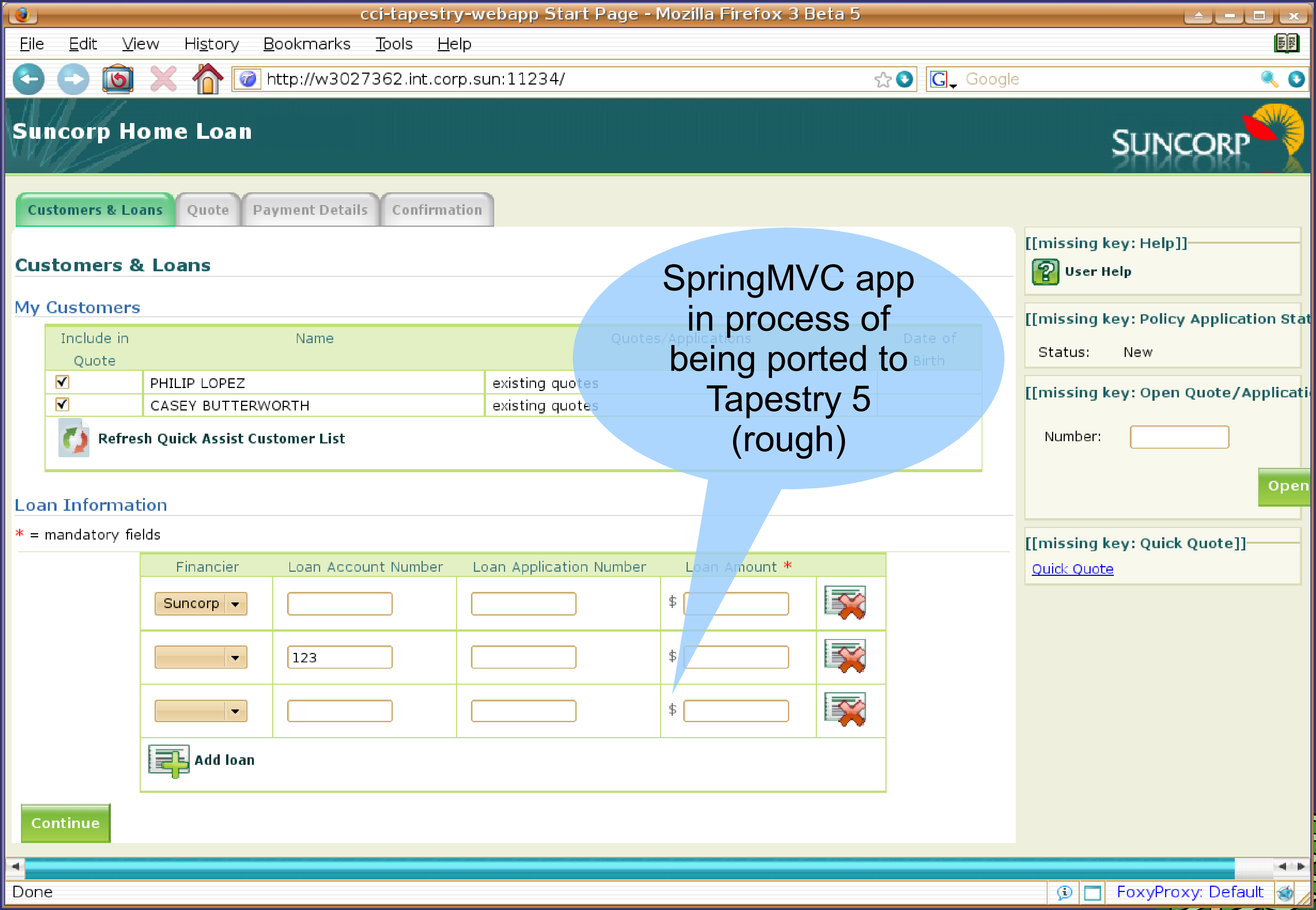


Action requests – where is this code?

- MVC approaches are ‘action’ oriented
  - Code is found in ‘action’ controllers
  - Separate from the ‘page’ model/view
- Component approaches provide a ‘presentation model’
  - The ‘event listener’ code for an action is found in the component (page) class that was responsible for rendering the action URL
    - e.g., action link, form ‘action’
- Action typically returns a ‘render’ response
  - POST+Redirect+GET a good strategy



Anatomy of an *componentised* application



SpringMVC app in process of being ported to Tapestry 5 (rough)

cci-tapestry-webapp Start Page - Mozilla Firefox 3 Beta 5

File Edit View History Bookmarks Tools Help

http://w3027362.int.corp.sun:11234/

Suncorp Home Loan

Customers & Loans | Quote | Payment Details | Confirmation

Customers & Loans

My Customers

Include in Quote	Name	Quotes/Applications	Date of Birth
✓	PHILIP LOPEZ	existing quotes	
✓	CASEY BUTTERWORTH	existing quotes	

Refresh Quick Assist Customer List

Loan Information

* = mandatory fields

Financier	Loan Account Number	Loan Application Number	Loan Amount *	
Suncorp			$	
	123		$	
			$	

Add loan

Continue

[[missing key: Help]]

User Help

[[missing key: Policy Application Stat

Status: New

[[missing key: Open Quote/Applicati

Number:

Open

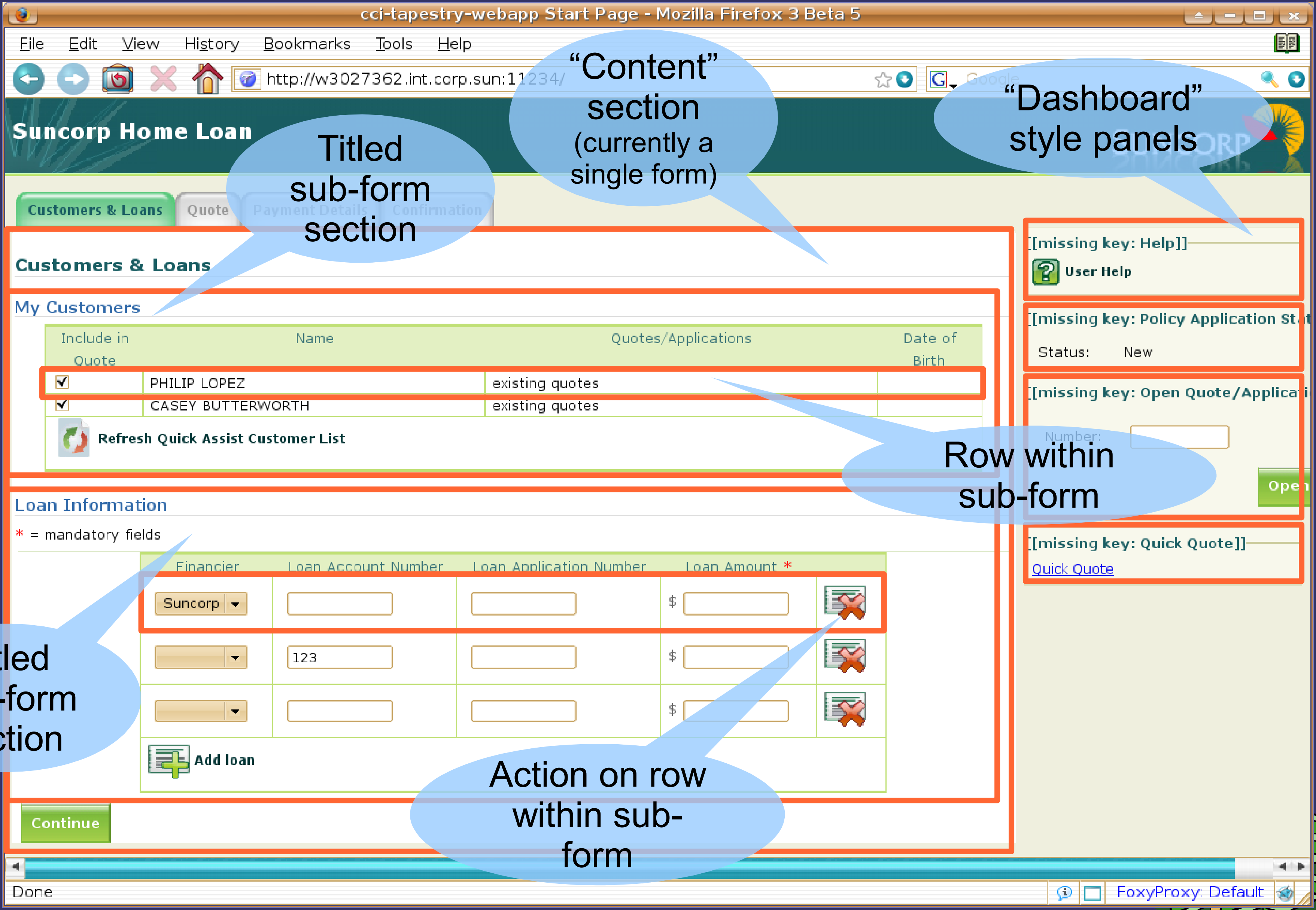
[[missing key: Quick Quote]]

Quick Quote

Done

FoxyProxy: Default

Anatomy of an *componentised* application


cci-tapestry-webapp Start Page - Mozilla Firefox 3 Beta 5
File Edit View History Bookmarks Tools Help
http://w3027362.int.corp.sun:11234/
Suncorp Home Loan
Customers & Loans
Quote
Payment Details
Confirmation
"Content" section (currently a single form)
"Dashboard" style panels
Titled sub-form section
Row within sub-form
Titled sub-form section
Action on row within sub-form
[missing key: Help]]
User Help
[missing key: Policy Application St
Status: New
[missing key: Open Quote/Applicati
Number:
Open
[missing key: Quick Quote]]
Quick Quote
Continue
Done
FoxyProxy: Default

Customers & Loans

My Customers

Include in Quote	Name	Quotes/Applications	Date of Birth
✓	PHILIP LOPEZ	existing quotes	
✓	CASEY BUTTERWORTH	existing quotes	

Refresh Quick Assist Customer List

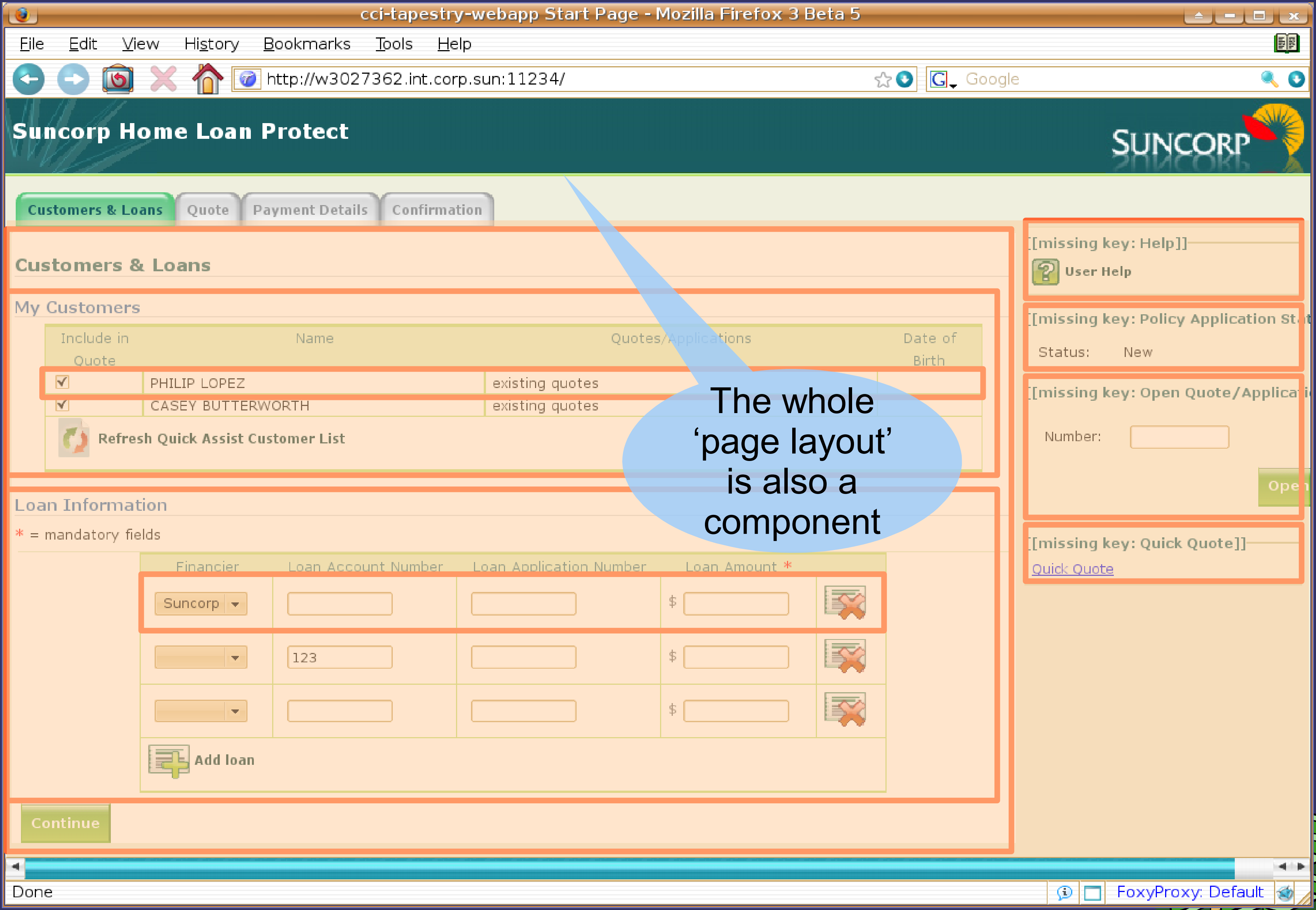
Loan Information

* = mandatory fields

Financier	Loan Account Number	Loan Application Number	Loan Amount *	
Suncorp			$	
	123		$	
			$	

Add loan

Anatomy of an *componentised* application



cci-tapestry-webapp Start Page - Mozilla Firefox 3 Beta 5

http://w3027362.int.corp.sun:11234/

Suncorp Home Loan Protect

Customers & Loans | Quote | Payment Details | Confirmation

Customers & Loans

My Customers

Include in Quote	Name	Quotes/Applications	Date of Birth
✓	PHILIP LOPEZ	existing quotes	
✓	CASEY BUTTERWORTH	existing quotes	

Refresh Quick Assist Customer List

Loan Information

* = mandatory fields

Financier	Loan Account Number	Loan Application Number	Loan Amount *	
Suncorp			$	
	123		$	
			$	

Add loan

Continue

[missing key: Help]]

User Help

[missing key: Policy Application St

Status: New

[missing key: Open Quote/Applica

Number:

Ope

[missing key: Quick Quote]]

Quick Quote

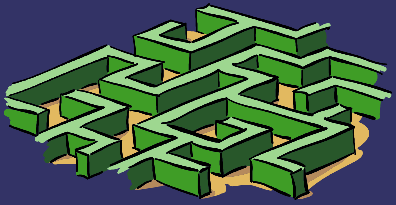
Done

FoxyProxy: Default

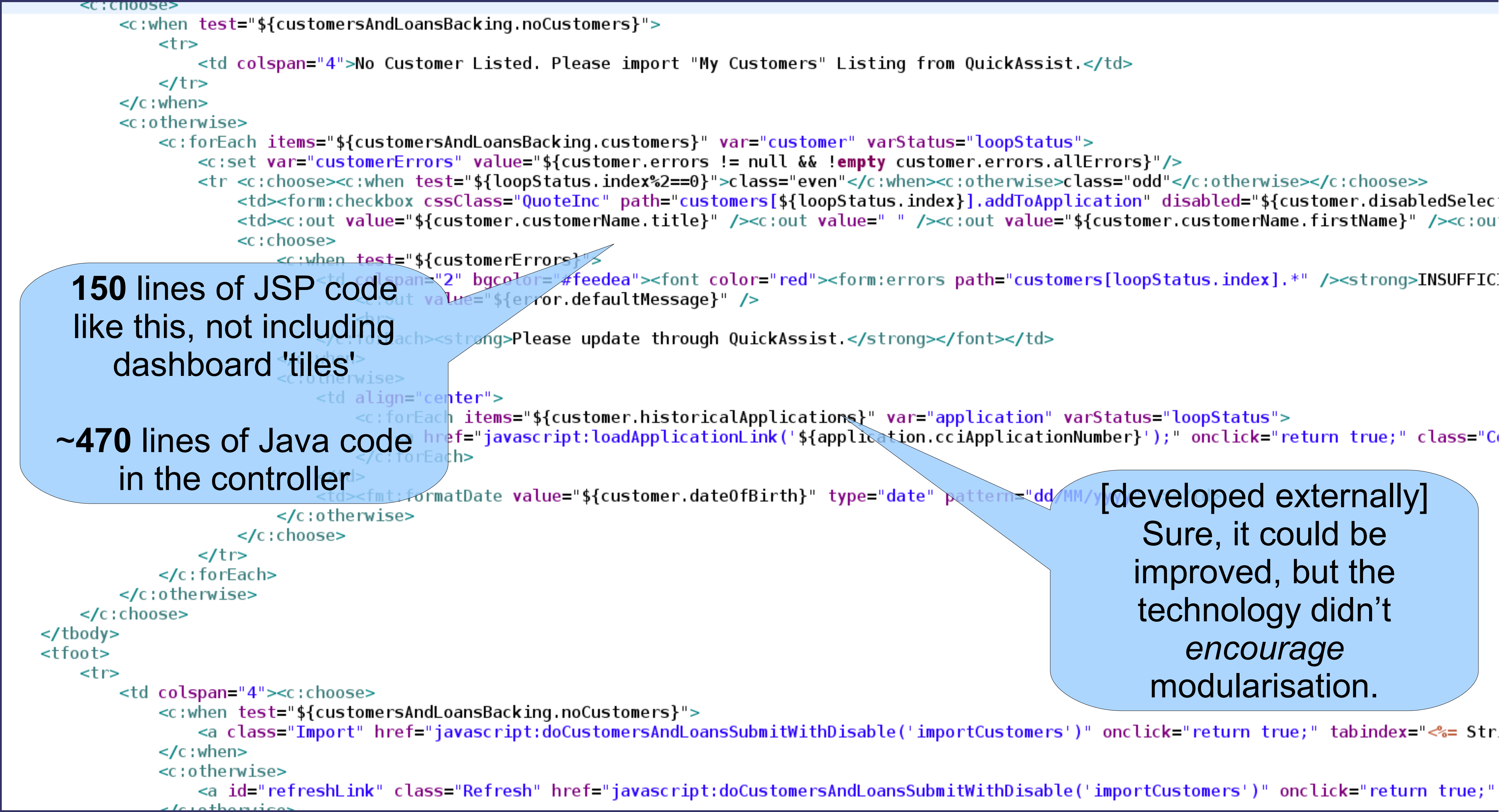
The whole 'page layout' is also a component

Some examples of componentisation

- Porting the previous app from SpringMVC
- Web ‘standards’... as components
- More sophisticated reusable UI components



The SpringMVC version, *as developed*



```
        <c:choose>
            <c:when test="${customersAndLoansBacking.noCustomers}">
                <tr>
                    <td colspan="4">No Customer Listed. Please import "My Customers" Listing from QuickAssist.</td>
                </tr>
            </c:when>
            <c:otherwise>
                <c:forEach items="${customersAndLoansBacking.customers}" var="customer" varStatus="loopStatus">
                    <c:set var="customerErrors" value="${customer.errors != null && !empty customer.errors.allErrors}"/>
                    <tr <c:choose><c:when test="${loopStatus.index%2==0}">class="even"</c:when><c:otherwise>class="odd"</c:otherwise></c:choose>>
                        <td><form:checkbox cssClass="QuoteInc" path="customers[${loopStatus.index}].addToApplication" disabled="${customer.disabledSelec
                        <td><c:out value="${customer.customerName.title}" /><c:out value=" " /><c:out value="${customer.customerName.firstName}" /><c:ou
                        <c:choose>
                            <c:when test="${customerErrors}">
                                <td colspan="2" bgcolor="#feedea"><font color="red"><form:errors path="customers[loopStatus.index].*" /><strong>INSUFFIC
                                    <c:out value="${error.defaultMessage}" />
                                    <br>
                                </c:forEach><strong>Please update through QuickAssist.</strong></font></td>
                            </c:when>
                            <c:otherwise>
                                <td align="center">
                                    <c:forEach items="${customer.historicalApplications}" var="application" varStatus="loopStatus">
                                        <a href="javascript:loadApplicationLink('${application.cciApplicationNumber}');" onclick="return true;" class="C
                                    </c:forEach>
                                </td>
                                <td><fmt:formatDate value="${customer.dateOfBirth}" type="date" pattern="dd/MM/y
                            </c:otherwise>
                        </c:choose>
                    </tr>
                </c:forEach>
            </c:otherwise>
        </c:choose>
    </tbody>
    <tfoot>
        <tr>
            <td colspan="4"><c:choose>
                <c:when test="${customersAndLoansBacking.noCustomers}">
                    <a class="Import" href="javascript:doCustomersAndLoansSubmitWithDisable('importCustomers')" onclick="return true;" tabindex="<%= Str
                </c:when>
                <c:otherwise>
                    <a id="refreshLink" class="Refresh" href="javascript:doCustomersAndLoansSubmitWithDisable('importCustomers')" onclick="return true;"
                </c:otherwise>
```

150 lines of JSP code like this, not including dashboard 'tiles'

~**470** lines of Java code in the controller

[developed externally] Sure, it could be improved, but the technology didn't *encourage* modularisation.

Porting to Tapestry 5 (trial, in progress)

```
<t:layout.PageLayout xmlns:t="...">
    <t:parameter name="content">
        <t:customer.CustomersAndLoansPanel/>
    </t:parameter>
</t:layout.PageLayout>
```

```
<t:container xmlns:t="...">
    <h3>Customers &amp; Loans</h3>

    <t:form t:id="customersAndLoansForm">
        <t:errors />
        <h4>My Customers</h4>
        <t:customer.QuickAssistCustomersPanel />

        <h4>Loan Information</h4>
        <t:customer.AddLoansPanel />

        <t:submit
            t:id="continue"
            value="Continue"
            class="greenPositive"
            onmouseover="this.className='greenPositive greenHover'"
            onmouseout="this.className='greenPositive'" />
    </t:form>
</t:container>
```

Porting to Tapestry 5 (trial, in progress)

```
<t:container xmlns:t="http://tapestry.apache.org/schema/tapestry_5_0_0.xsd">
    <fieldset class="QuickCal">
        <div>
            <table class="QuickCal">
                <colgroup>
                    <col id="QuoteInclude" />
                    <col id="Name" />
                    <col id="Quotes" />
                    <col id="DOB" />
                </colgroup>
                <thead>
                    <tr>
                        <th scope="col">Include in Quote</th>
                        <th scope="col">Name</th>
                        <th scope="col">Quotes/Applications</th>
                        <th scope="col">Date of Birth</th>
                    </tr>
                </thead>
                <tbody>
                    <t:if test="customersAvailable">
                        <tr
                            t:type="loop"
                            source="customers"
                            encoder="customerRowIdEncoder"
                            value="customer">
                            <t:customer.QuickAssistCustomerRow customer="customer" />
                        </tr>
                        <t:parameter name="else">
                            <tr>
                                <td colspan="4">
                                    No Customer Listed. Please import "My Customers" Listing from
                                    QuickAssist.
                                </td>
                            </tr>
                        </t:parameter>
                    </t:if>
                </tbody>
                <tfoot>
                    <tr>
                        <td colspan="4">
                            <t:common.SubmitLink
                                t:id="refreshCustomerListLink"
                                buttonClass="Refresh"
                                label="Refresh Quick Assist Customer List" />
                        </td>
                    </tr>
                </tfoot>
            </table>
        </div>
    </fieldset>
</t:container>
```

Less than 50 lines in this file

~40 lines in the page class

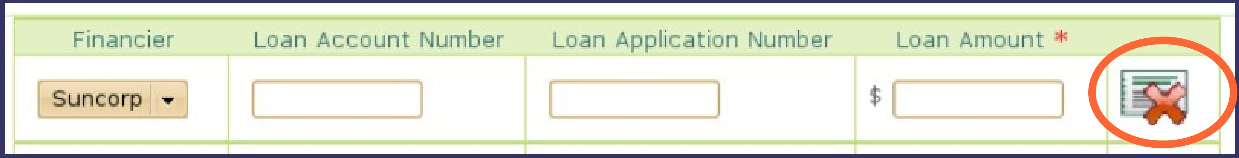
(albeit not quite feature complete)

T5 loop component, with *robust* support for forms.

An page-specific panel, with a type-safe parameter.

Reusable 'widget' - encapsulate implementation.

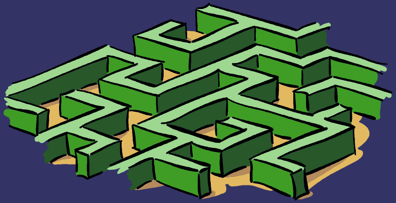
Deleting a row... (one) SpringMVC style



Financier	Loan Account Number	Loan Application Number	Loan Amount *	
Suncorp			$	

- Set **operation** and **row number** as hidden fields for operation in form (using JS) and submit form.
- In the controller's onSubmit:

```
if ("deleteLoan".equals(operation)) {
    backing.getLoans().remove(Integer.parseInt(request.getParameter("rownumber")));
}
return new ModelAndView(new RedirectView(getSuccessView()));
```

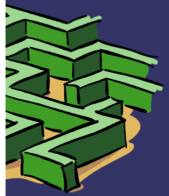


Deleting a row... (one) Wicket style

- Listener is ‘embedded’ in component tree that was rendered
  - Uses session and Serializable LoanDetail

```
private ListView createLoanListView()
{
    return new ListView("loansList", new PropertyModel(this, "loans")) {
        protected void populateItem(final ListItem item)
        {
            final LoanDetail loan = (LoanDetail) item.getModelObject();
            // ...
            // item.add(makeTextField(loan, "amount"));
            item.add(buildDeleteLoanRecordSubmitLink(loan));
        }
    };
}
```

```
private SubmitLink buildDeleteLoanRecordSubmitLink(final LoanDetail loan)
{
    return new SubmitLink("deleteLoanRecord") {
        public void onSubmit()
        {
            loans.remove(loan);
        }
    };
}
```



Deleting a row... (one) Tapestry 5 style

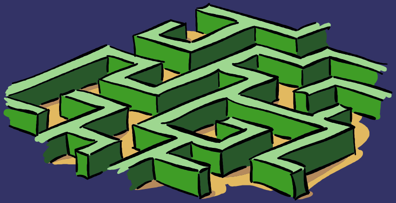
- Add an action link with a **context**
  - Encode 'primary key' to client (HTML)

```
<t:if test="showDelete">
    <a t:id="deleteLoanRecord" class="DeleteRecord" />
    <t:parameter name="else">
        <img src="${spacerImageUrl}" height="32" width="32" />
    </t:parameter>
</t:if>
```

```
@Component(parameters = { "context=loanDetail.rowId", "event=deleteLoanRecord" })
private EventLink deleteLoanRecord;
```

```
void onDeleteLoanRecord(String rowId)
{
    loanDetails.remove(indexOfLoan(UUID.fromString(rowId)));
}
```

Using a UUID is a robust (custom) row identification approach... but **ugly**, and could be extracted out.



Codifying web development standards

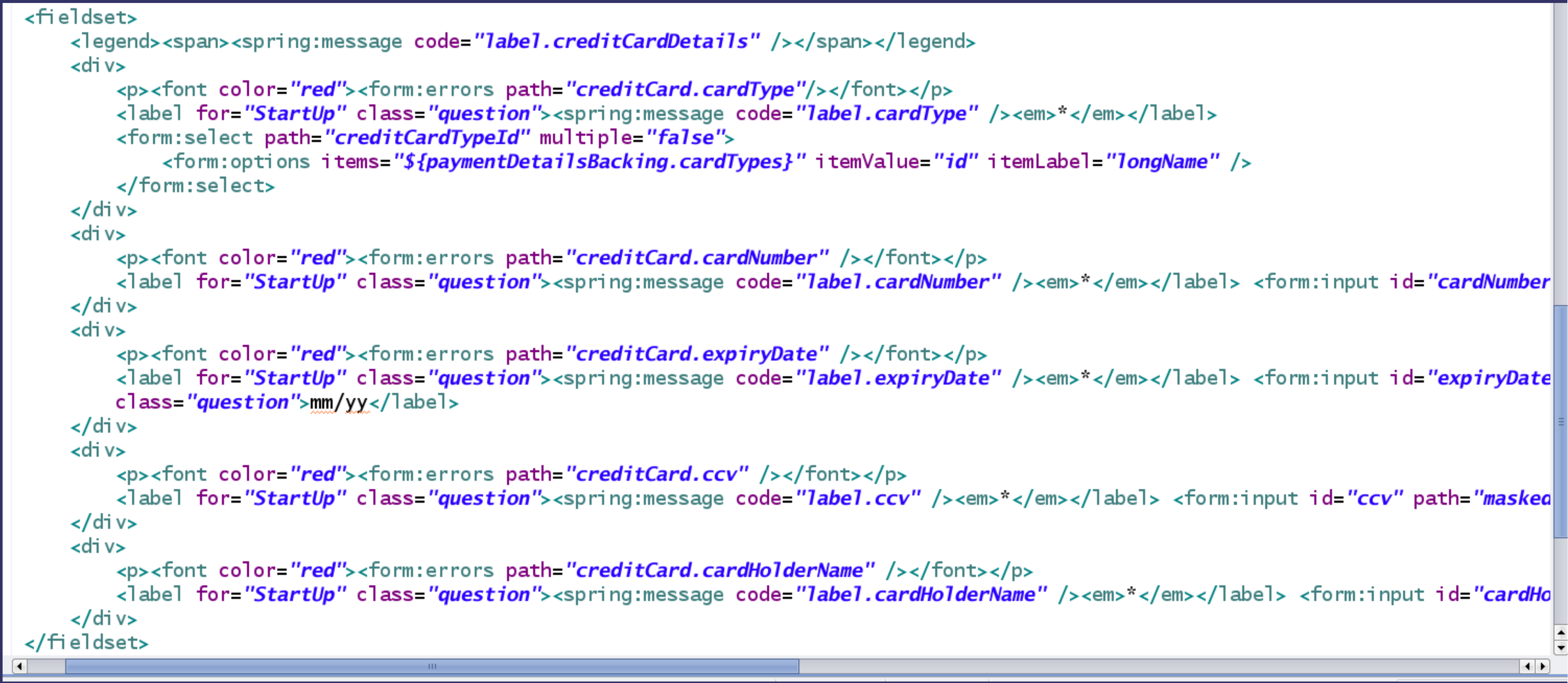
- Struggle to maintain corporate web L&F standards – many violations of DRY
  - CSS requires boilerplate HTML markup

```
<div>
  <p>
    <font color="red">
      <form:errors path="creditCard.cardNumber" />
    </font>
  </p>
  <label for="StartUp" class="question">
    <spring:message code="label.cardNumber" />
    <em>*</em>
  </label>
  <form:input id="cardNumber" path="maskedCreditCardNumber"
      size="16" maxlength="16" onfocus="this.select();" />
</div>
```

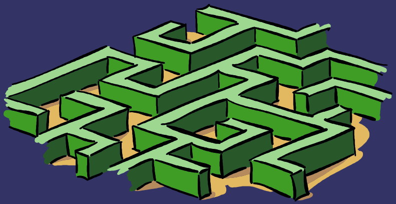
Copy and paste errors!

About the only interesting thing!

So you end up with...



```
<fieldset>
    <legend><span><spring:message code="label.creditCardDetails" /></span></legend>
    <div>
        <p><font color="red"><form:errors path="creditCard.cardType"/></font></p>
        <label for="StartUp" class="question"><spring:message code="label.cardType" /><em>*</em></label>
        <form:select path="creditCardTypeId" multiple="false">
            <form:options items="${paymentDetailsBacking.cardTypes}" itemValue="id" itemLabel="longName" />
        </form:select>
    </div>
    <div>
        <p><font color="red"><form:errors path="creditCard.cardNumber" /></font></p>
        <label for="StartUp" class="question"><spring:message code="label.cardNumber" /><em>*</em></label> <form:input id="cardNumber
    </div>
    <div>
        <p><font color="red"><form:errors path="creditCard.expiryDate" /></font></p>
        <label for="StartUp" class="question"><spring:message code="label.expiryDate" /><em>*</em></label> <form:input id="expiryDate
        class="question">mm/yy</label>
    </div>
    <div>
        <p><font color="red"><form:errors path="creditCard.ccv" /></font></p>
        <label for="StartUp" class="question"><spring:message code="label.ccv" /><em>*</em></label> <form:input id="ccv" path="maskec
    </div>
    <div>
        <p><font color="red"><form:errors path="creditCard.cardHolderName" /></font></p>
        <label for="StartUp" class="question"><spring:message code="label.cardHolderName" /><em>*</em></label> <form:input id="cardHo
    </div>
</fieldset>
```



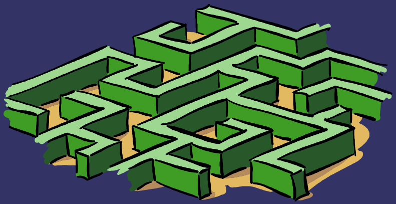
Simple components offer a solution

```
<t:container xmlns:t="http://tapestry.apache.org/schema/tapestry_5_0_0.xsd">
    <t:scform.QuestionTemplate for="cardType">
        <t:radioGroup t:id="cardType">
            <t:loop t:id="cardTypeLoop">
                <t:radio t:id="currentCardTypeRadio" />
                <t:label for="currentCardTypeRadio" />
            </t:loop>
        </t:radioGroup>
    </t:scform.QuestionTemplate>
    <t:scform.QuestionTemplate for="cardNumber">
        <input t:id="cardNumber" />
    </t:scform.QuestionTemplate>
    <t:scform.MultiFieldQuestionTemplate t:id="cardExpirationDate" required="true">
        <input t:id="expirationMonth" size="2" />
        /
        <input t:id="expirationYear" size="2" />
    </t:scform.MultiFieldQuestionTemplate>
    <t:scform.QuestionTemplate for="cardSecurityCode">
        <input t:id="cardSecurityCode" />
    </t:scform.QuestionTemplate>
    <t:scform.QuestionTemplate for="cardholderName">
        <input t:id="cardholderName" />
    </t:scform.QuestionTemplate>
</t:container>
```

QuestionTemplate is a reusable component that encapsulates our (versioned) web development standards.

scform is the virtual package (namespace) for the Suncorp common form component library. Just drop-in the JAR.

Repetition here due to Tapestry restrictions (component encapsulation)



The QuestionTemplate component

```
@IncludeStylesheet("FormStyles.css")
public class QuestionTemplate
{
    @Parameter(name = "for", required = true, defaultPrefix = "component")
    private Field field;

    public Field getField()
    {
        return field;
    }

    public boolean isRequired()
    {
        return field.isRequired();
    }
}
```

This annotation **contributes** FormStyles.css to the document's HEAD (uniquely)

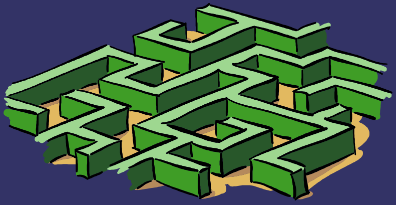
D.R.Y. – HTML markup is based on declarative validation on field.

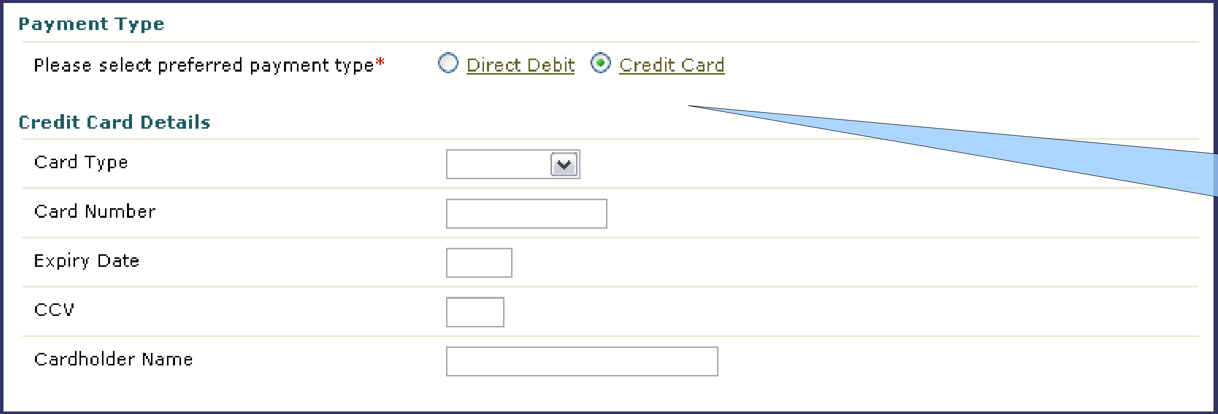
All the rest is boilerplate. The component expects a Field in its body, and renders it out here.

```
<t:container xmlns:t="http://tapestry.apache.org/schema/tapestry_5_0_0.xsd">
    <div class="scform-Question">
        <t:label t:id="label" class="scform-QuestionPrompt"
            for="prop:field" />
        <t:body />
        <t:if test="required">
            <span class="scform-RequiredIndicator">*</span>
        </t:if>
    </div>
</t:container>
```

A “Receive Payment” widget

- A few options for reuse – not in conflict
  - (existing) Credit card validation library (JAR, d.i.)
  - (existing) Payment service [SOA]
  - Perhaps a shared payment “web application”
    - **<u>But</u>** now requires application integration - fragile
    - Breaks user experience (e.g., multi-branding)
- Developing a robust page takes time
  - Lots of validations
  - Some dynamic behaviour
  - Reuse across many apps can save $$$
- DRY – use metadata from services
  - e.g. available credit card types





Payment Type

Please select preferred payment type*	○ Direct Debit [✓ Credit Card]

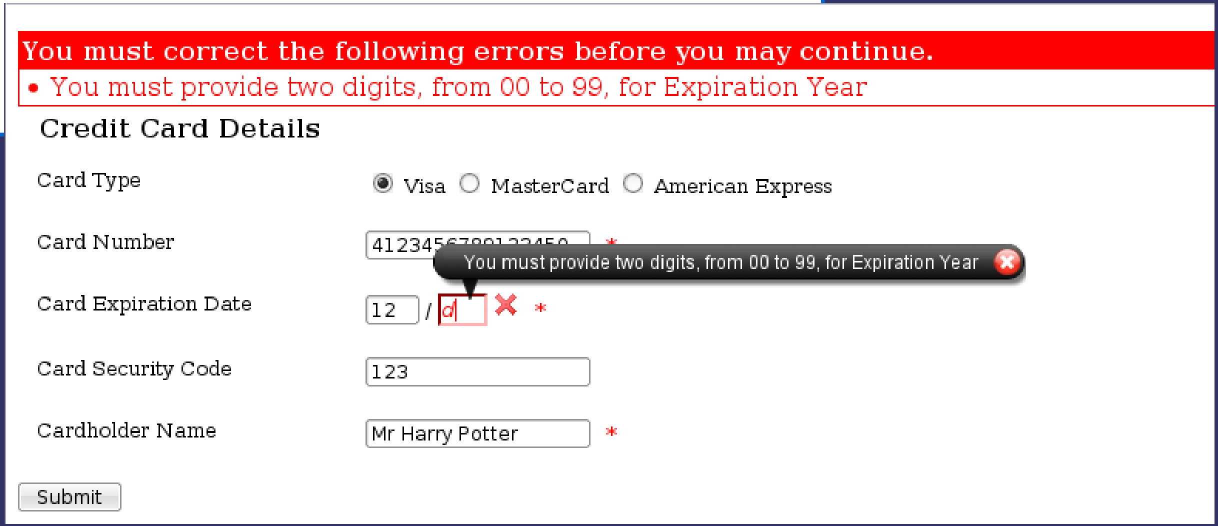
Credit Card Details

Card Type	
Card Number	
Expiry Date	
CCV	
Cardholder Name	

Lots of variations of this...

```
<html xmlns:t="http://tapestry.apache.org/schema/tapestry_5_0_0.xsd">
    <head>
        <title>Credit Card Form</title>
    </head>
    <body>
        <t:scform.SimpleForm>
            <t:scform.FormSection title="literal:Credit Card Details">
                <div t:id="creditCardPaymentDetails" />
            </t:scform.FormSection>
            <t:submit value="Submit" />
        </t:scform.SimpleForm>
    </body>
</html>
```

The beginnings of a shared component – a credit card details sub-form.



You must correct the following errors before you may continue.

- You must provide two digits, from 00 to 99, for Expiration Year

Credit Card Details

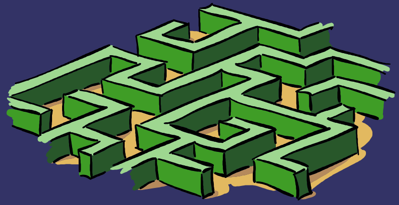
Card Type	[✓ Visa] ○ MasterCard ○ American Express
Card Number	412345 *
Card Expiration Date	12 / d *
Card Security Code	123
Cardholder Name	Mr Harry Potter *

You must provide two digits, from 00 to 99, for Expiration Year

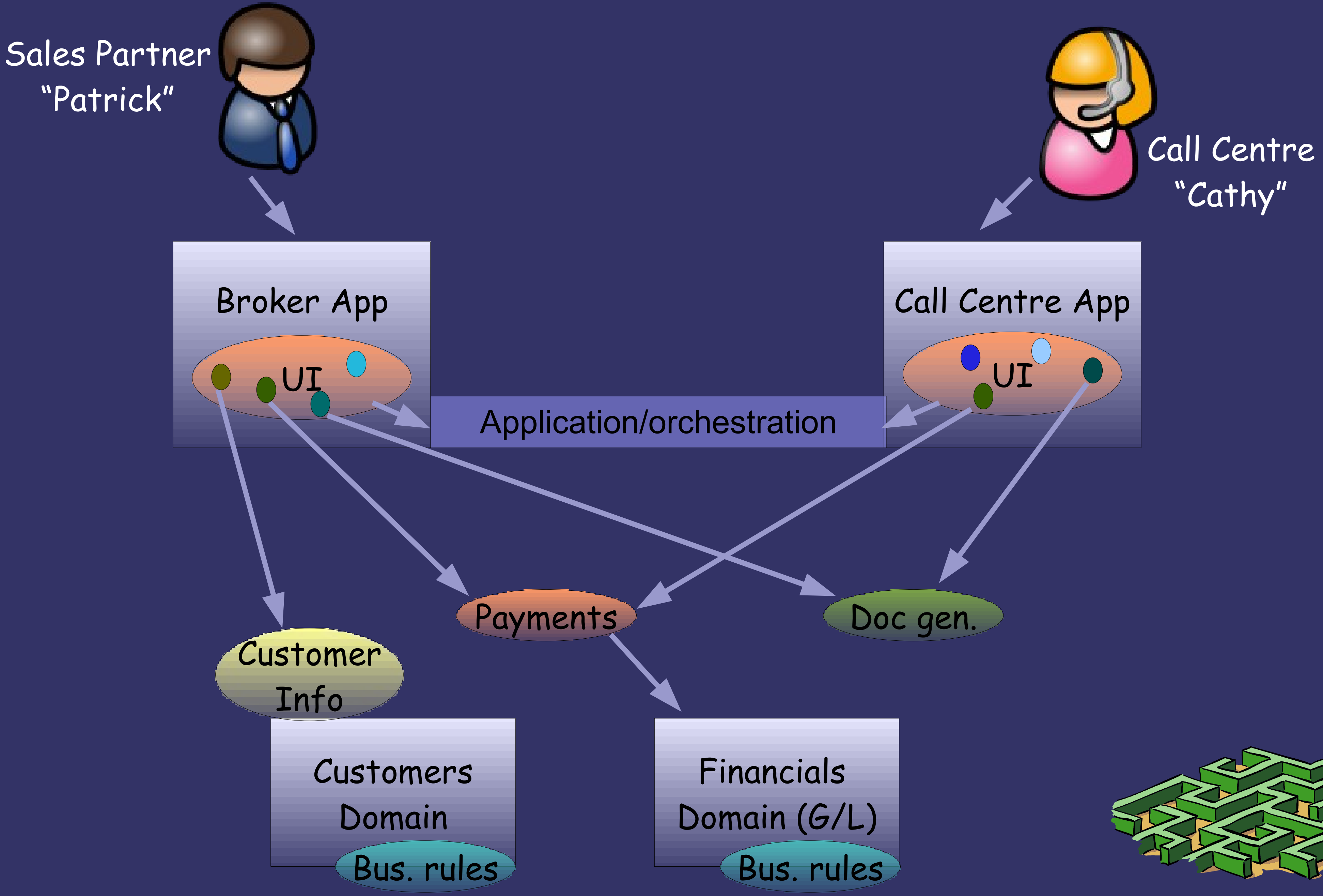
Submit

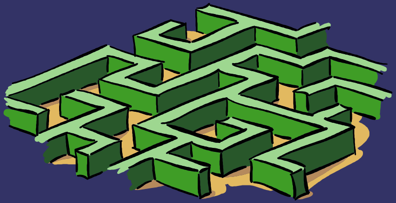
Architectural directions

- Toward true ‘composite’ applications and multiple **layers of reuse**
  - Shared services (business and utility)
  - Shared base widgets
  - Shared business-functionality widgets
- **Organisation role-centric** applications
- Service-oriented system ‘product’ mgmt
- Declarative approaches (higher abstraction)
  - Model-driven benefits, but not the code-gen...
  - Convention-over-configuration



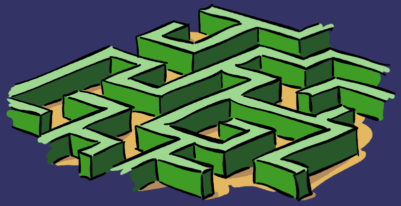
‘Future’ application portfolio





Challenges to reuse in the web app layer

- Multiple languages/technologies
  - JavaScript and DHTML is common, so ‘widgets’ can start there (but needs JavaScript)
  - Can couple client-side widgets to server-side resources using server-side frameworks
- Does the presentation layers change too quickly to make reuse warranted?
  - Gains are achievable at the enterprise scale...



Take-home messages...

- We often violate the “Don’t Repeat Yourself” principle in web app development
  - Boilerplate HTML within an application
  - Boilerplate HTML between applications
  - Similar functionality between applications
- Component-based frameworks help...
  - Natural approach to modularisation
  - Enables component reuse between applications
- There’s a learning curve, and more “magic”, so choose carefully...
  - Clean, correct abstractions
  - Long-term productivity most important?