



The Grantlee Template System



Stephen Kelly



What is Grantlee

How does it work?

How will applications use it?



The old way

```
QString str;  
str.append("<html><h1>" + title + "</h1>");  
str.append("<div>" + content + "</div>");  
str.append("</html>");
```



The old way

```
QString str;  
str.append("<html><h1>" + title + "</h1>");  
str.append("<div>" + content + "</div>");  
str.append("</html>");
```

Goals

- Separate exported data from its presentation
- Simplify theming for application developers
- Use syntax of the Django Template system



The new way

```
TemplateLoader *tl = TemplateLoader::instance();  
tl->setTheme(themeName);  
Template* t = tl->getByName("template.html")  
contextMap.insert("entities", selectedEntities);  
Context c(contextMap);  
return t->render(&c);
```



The new way

```
<html>
{# A simple comment #}
<div>{% for entity in entities %}
  {% if entity.isBook %}
    <h1>{{ entity.title }}</h1>
    {% include "booktemplate.html" %}
  {% else %}
    {% include "pagetemplate.html" %}
  {% endif %}
{% endfor %}</div>
</html>
```



The Grantlee Template System



Demo



The Grantlee Template System



The screenshot shows the KJots application window titled "KJots rewrite app". The interface includes a menu bar with "Settings" and "Help", and a toolbar with "Change Theme" and "Export".

Left Panel (Tree View):

- Name
- Bar Book
 - + Bat Book
 - Bar Page
 - Bar2 Page
 - Bar4 Page
 - Bar3 Page
 - + Foo Book

Right Panel (Preview):

Bar Book

Bat Book

Bat Page
Bat File content

Bat2 Page
Bat2 File content

Bar Page
Bar File content

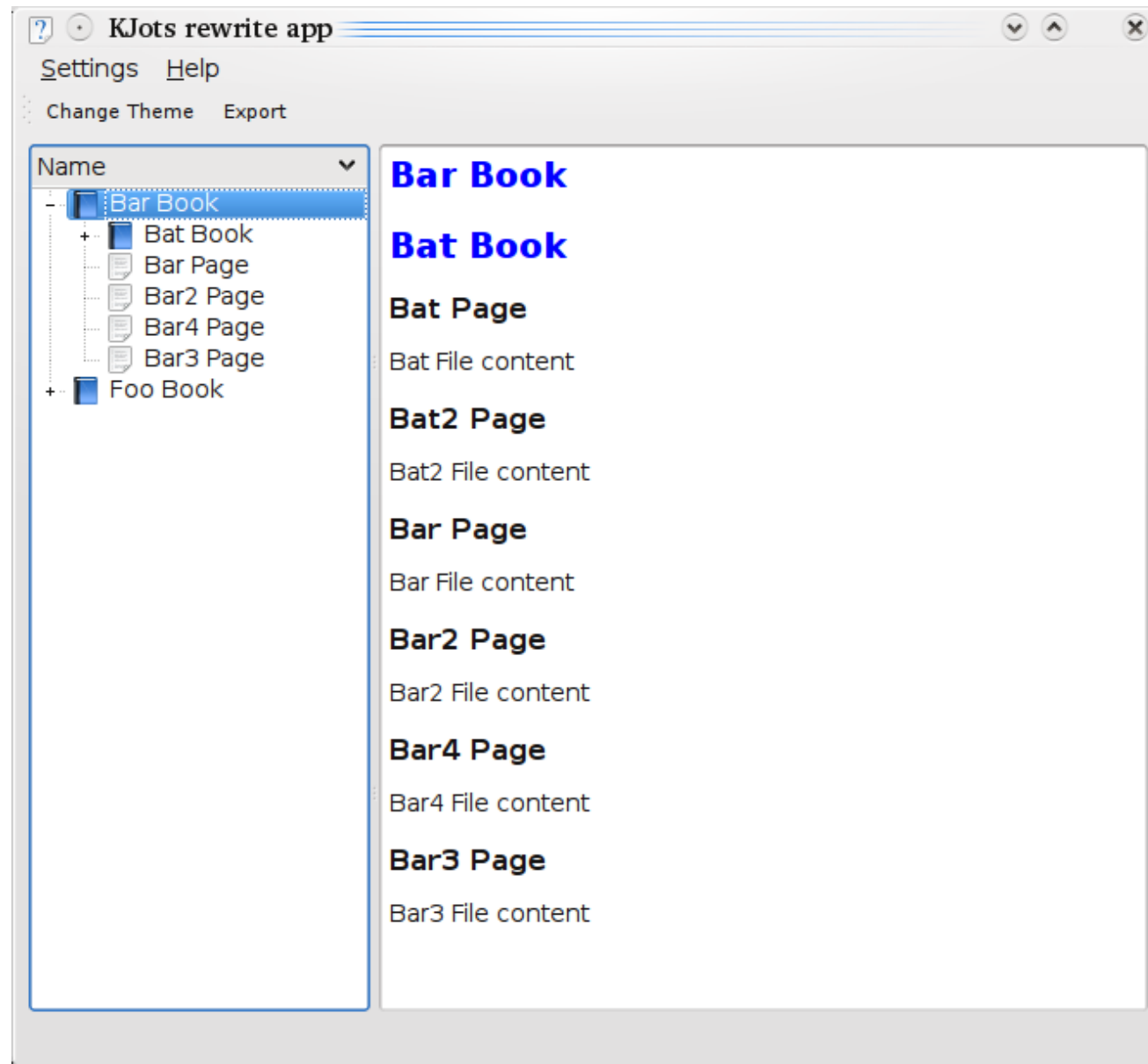
Bar2 Page
Bar2 File content

Bar4 Page
Bar4 File content

Bar3 Page
Bar3 File content



The Grantlee Template System





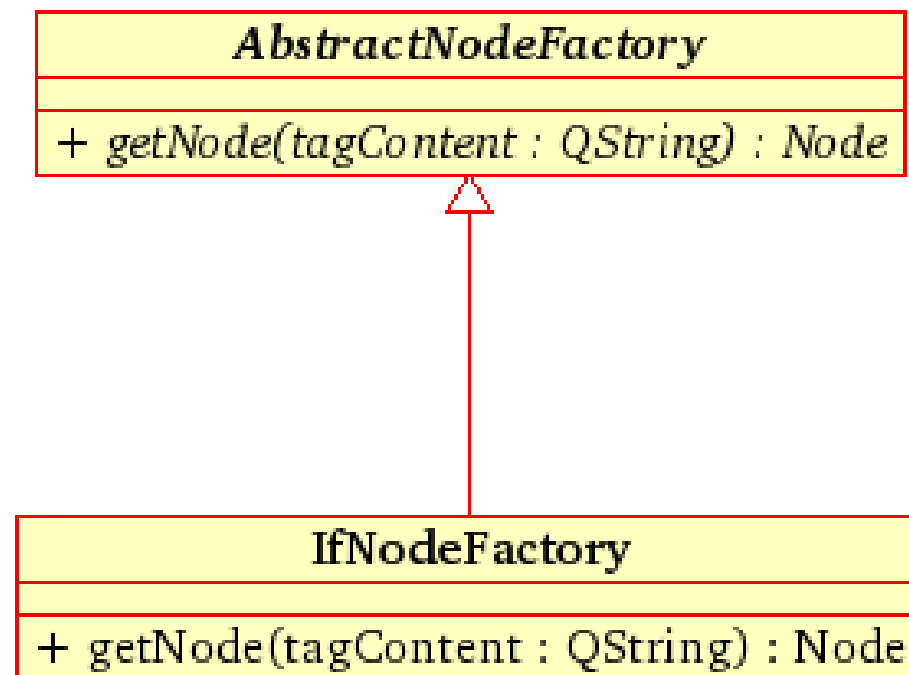
Tokens

- **Text:** “<html>”, etc
- **Comment:** “{# A simple comment #}”
- **Variable:** “{{ entity.title }}”
- **Tag:** “{% for entity in entities %}”,
“{% if entity.isBook %}”, etc



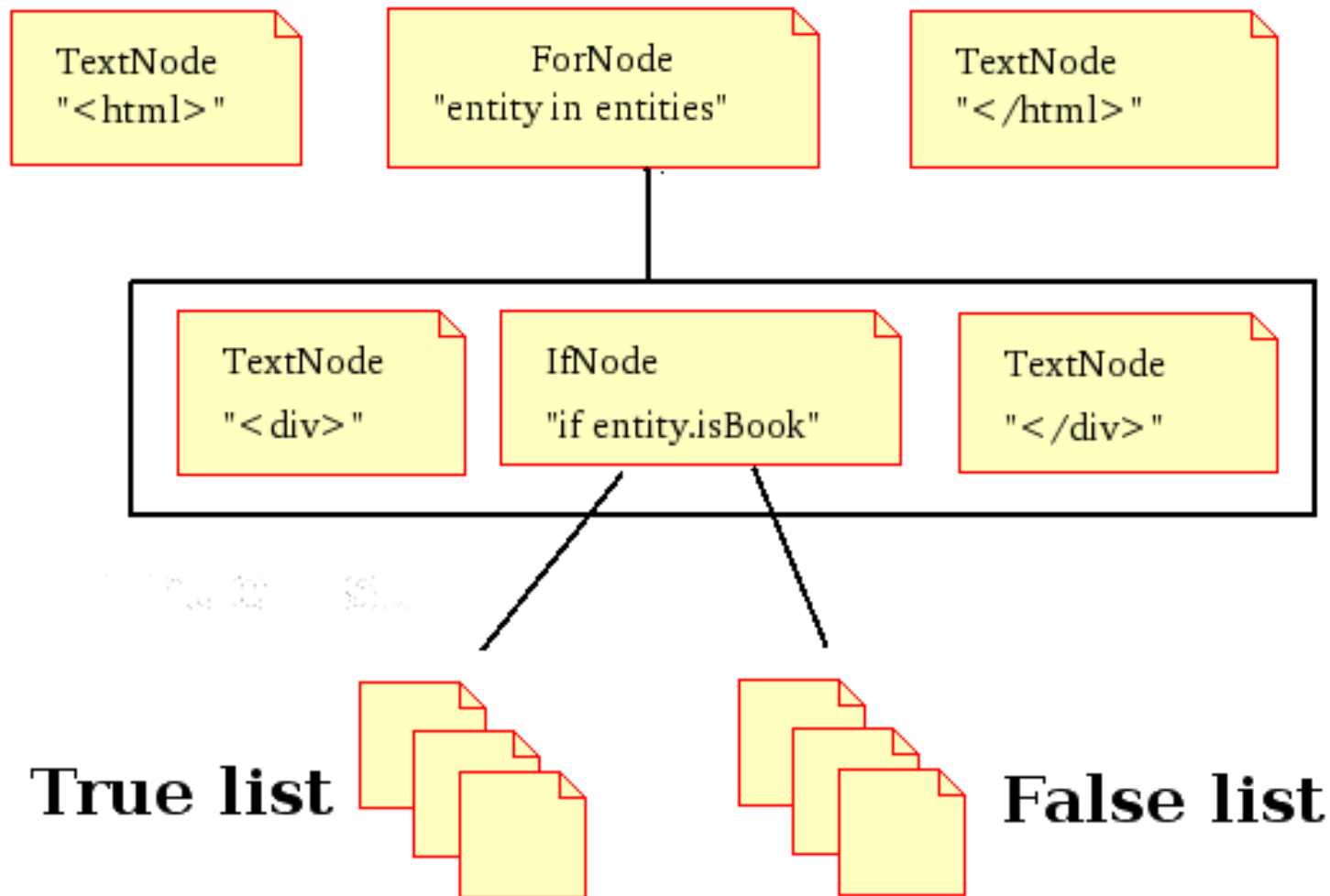
Nodes

- **Generated from Tag tokens**
- **Extensible using QtPlugin system**





Nodes





Context

- **Objects injected into the template**
 - **Must be introspectable**
 - **Type agnostic**
 - **Should be read-only**
 - **Methods should take no arguments**
 - **Possible to iterate over iterable types (lists, maps etc)**
- **This is all free in Python**



Context

```
class KJotsEntity : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QString title READ title)
    Q_PROPERTY(bool isBook READ isBook)
    Q_PROPERTY(bool isPage READ isPage)
    Q_PROPERTY(QString content READ content)
    Q_PROPERTY(QVariantList entities READ entities)

public:
    bool isBook();
    bool isPage();
    QString title();
    QString content();
    QVariantList entities();
};
```



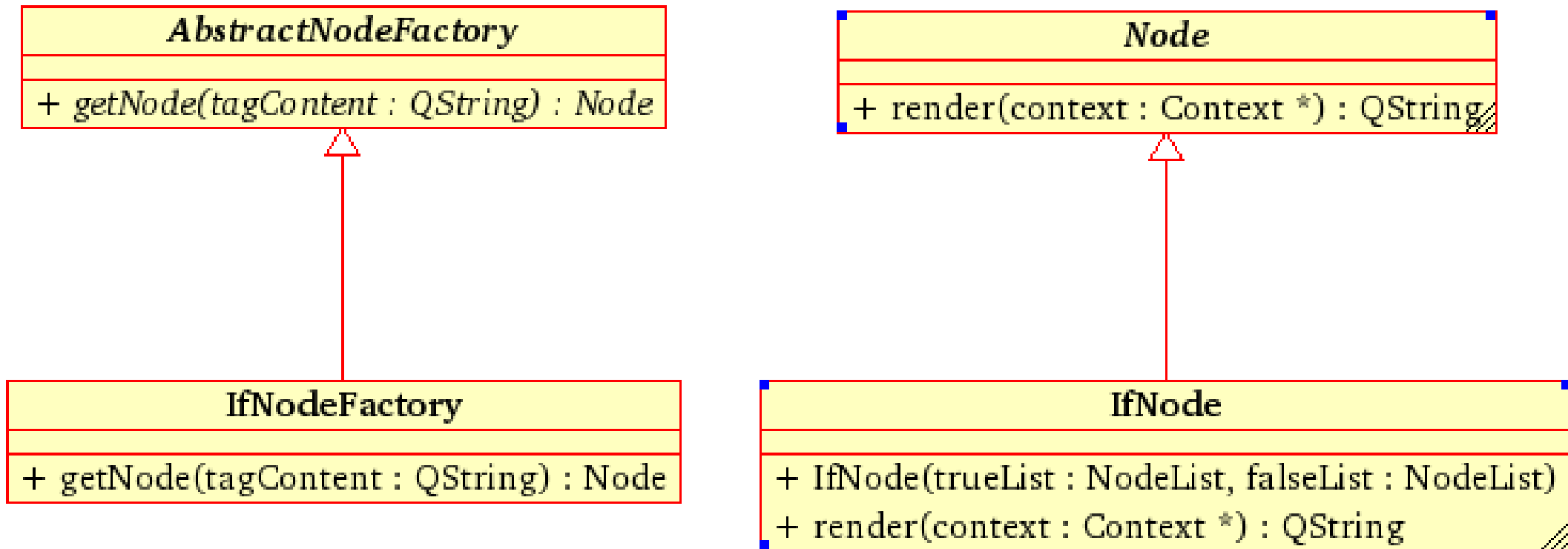
Rendering

```
TemplateLoader *tl = TemplateLoader::instance();
tl->setTheme(themeName);
Template *t = tl->getTemplate("template.html")
contextMap.insert("entities", selectedEntities);
Context c(contextMap);

// What does this do?
return t->render(&c);
```



Rendering





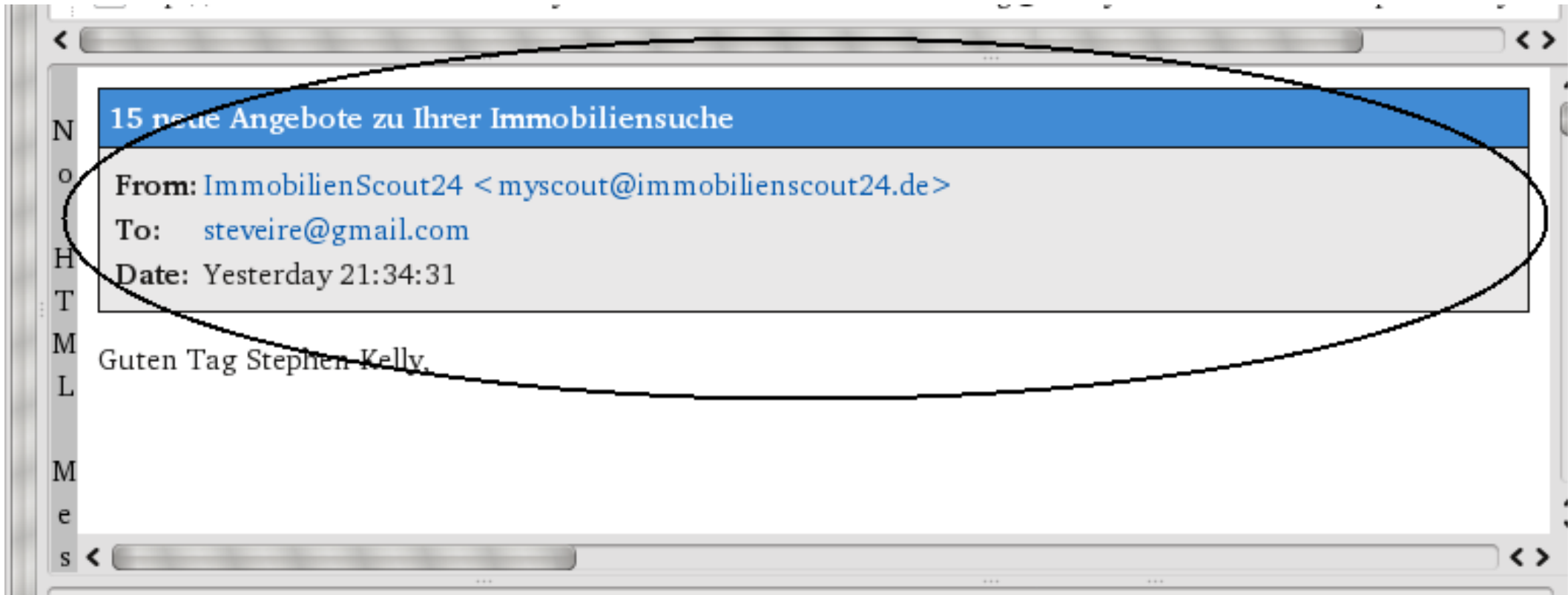
Rendering

- **Tries to resolve variables in the Context**
 - (eg, “entity.1.isPage”)
- **Nodes render child nodes and use the result**



Consumers

- Kjets, KMail, KNode, Akregator, etc





Consumers

- Kjots, KMail, KNode, Akregator, etc
- Complex MIME message structures?

To: {{ recipient }}

Date: {% now %}

Subject: {{ subject }}

{% for header in extraHeaders %}

{{ header.key }}: {{header.value }}

{% endfor %}

{% for part in message %}

{% include part.typeTemplate %}

{% endfor %}



Consumers

- KJots, KMail, KNode, Akregator, etc
- Complex MIME message structures?
- Akonadi mail-merge agent?

Dear `{{ customer.contactName }}`,

We are out of stock of `{{ product }}`, so your orders on
`{% for order in customer.affectedOrders %}`

`* {{ order.date }}`

`{% endfor %}`

Can not be filled.



Consumers

- **Kjots, KMail, KNode, Akregator, etc**
- **Complex MIME message structures?**
- **Akonadi mail-merge agent?**
- **Kexi reports?**
- **Kopete themes?**
- **Browser interface to embedded systems?**



Other

- **Templates can be 'inherited' and parts overridden**
- **Existing tags exist for ifequal, with, cycle etc**
- **Tags can be defined in QtScript**
- **Variables can be passed through filters**



TODO

- **Package structure and GHNS**
- **Redesign for progressive updates.**
- **Performance tests comparison with XSLT**
- **i18n**
- **Evaluate QScriptEngine more**



Questions & Answers

<http://docs.djangoproject.com/en/dev/topics/templates/>

gg: Django Template System