



Apache Isis



*Introducing...  
Apache Isis*



# Ubiquitous Language



*With a conscious effort by the team, the domain model can provide the backbone for that common language*

*Eric Evans,  
Domain Driven Design*



# What is Naked Objects ?

- An Architectural Pattern
  - automatically renders domain objects in an OOUI
  - fits in with the hexagonal architecture
- A Principle
  - all business functionality is encapsulated on the core business objects
  - “problem solver, not process follower”
- A natural bed-fellow for Domain-Driven Design
  - rapid prototyping & development





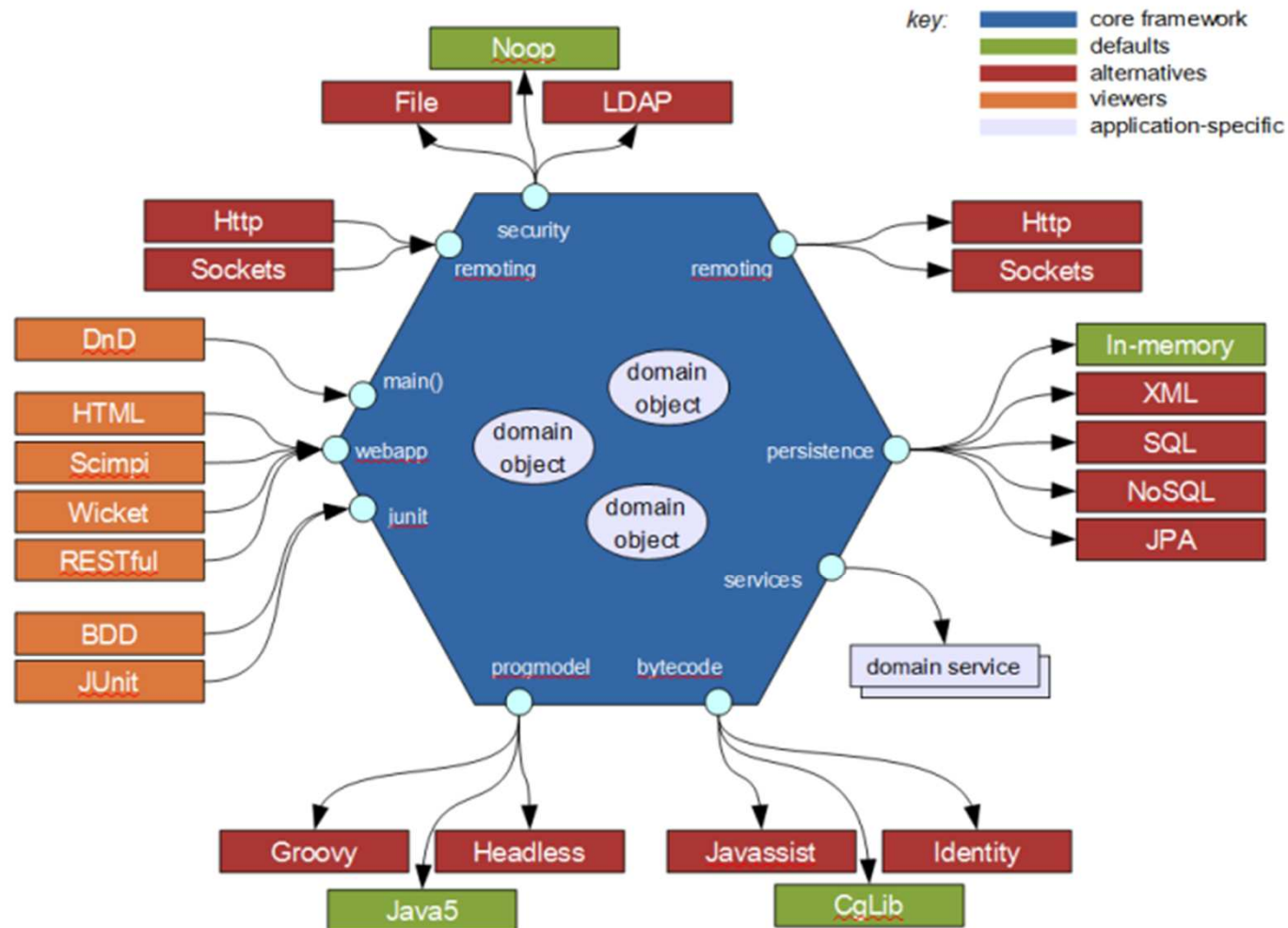
## An example of the DRY Principle

- The UI representations correspond directly with the underlying domain object model
- So, for instance:
  - objects instances exposed as icons
  - object properties / collections exposed in forms
  - object methods exposed as menu items
    - eg `Claim#submit(Approver)`
  - repositories/domain services exposed as desktop icons
    - eg `ClaimRepository`, `EmployeeRepository`

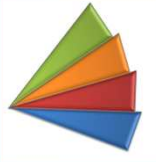




# What is Apache Isis?



Apache Isis combines the naked objects pattern  
with the hexagonal architecture



# Isis apps are just pojos

```
Claim.java x
package org.nakedobjects.examples.claims.dom.claim;

import java.util.ArrayList;

public class Claim extends AbstractDomainObject {

    // {{ Description
    private String description;
    @MemberOrder(sequence = "1")
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    // }}

    // {{ Date
    private Date date;
    @MemberOrder(sequence = "2")
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
    // }}

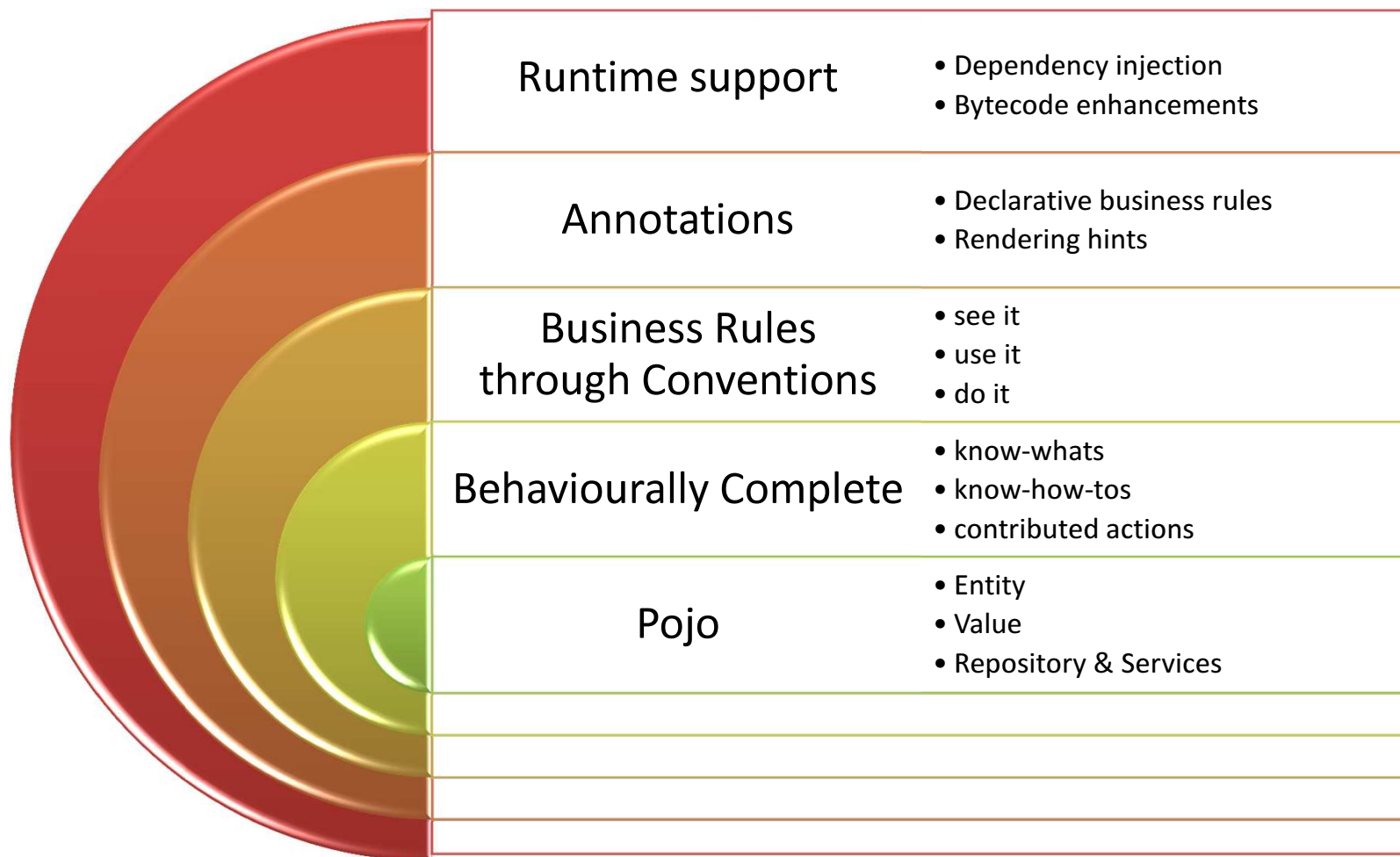
    // {{ Items
    private List<ClaimItem> items = new ArrayList<ClaimItem>();
    @MemberOrder(sequence = "6")
    public List<ClaimItem> getItems() {
        return items;
    }
    public void addToItems(ClaimItem item) {
        items.add(item);
    }
    // }}

    // {{ action: Submit
    public void submit(Approver approver) {
        setStatus("Submitted");
        setApprover(approver);
    }
    public String disableSubmit() {
        return getStatus().equals("New") ? null : "Claim has already been submitted";
    }
    public Object[] defaultSubmit() {
        return new Object[] { getClaimant().getApprover() };
    }
    // }}
}
```





# The Isis Programming Model





*So what does the app look like?*

- Let's see...







# Resources

- Apache Isis Incubator website
  - links to the mailing list (isis-dev), IRC (#apache-isis)
  - links the wiki, JIRA
  - available via Maven
  - <http://incubator.apache.org/isis>
- Richard Pawson's original thesis on Naked Objects
  - <http://incubator.apache.org/isis/Pawson-Naked-Objects-thesis.pdf>
- Dan Haywood's book
  - <http://www.pragprog.com/titles/dhnako>
- Naked Objects on .NET
  - <http://nakedobjects.net>

The Pragmatic  
Programmers

Domain-Driven Design  
Using Naked Objects





# The DSP: Why?

## Strategic Agility

Respond to  
unforeseen  
changes in  
business  
requirements

## Operational Agility

Provide  
clerical  
officers with  
greater  
flexibility to  
solve  
customers'  
problems

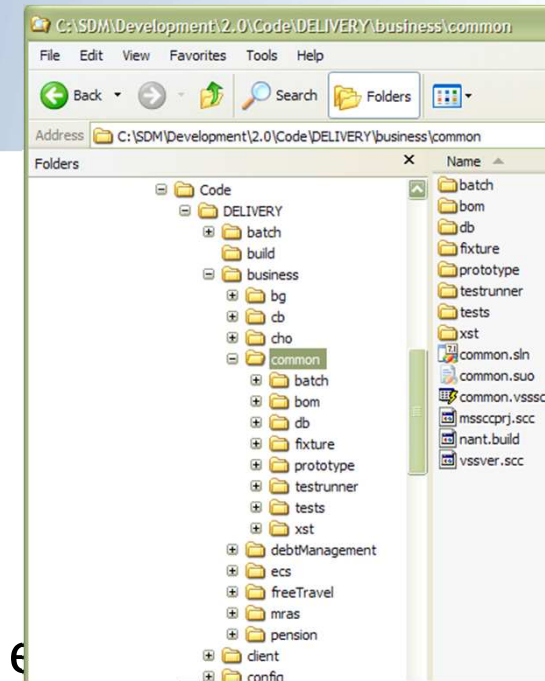
## Technical Agility

Accommodate  
changes in  
technology



# The DSP What?

- Platform for the future generation of business systems
  - the common BOM (a shared kernel)
  - a technology platform
    - UI, remoting, bespoke ORM, ...
- Specific applications replacing & extending e administration systems:
  - State pensions, Free Travel, Household Benefits, ECS, ...
  - Overpayment/Debt Management system, Medical Referrals, ...
- Integration with other systems, technologies and dept<sup>s</sup>
  - BizTalk messaging, batch, scanning, barcodes, ...
  - Central Printing, SMS, other media, ...





# Why the DSP's Naked Objects system makes for an interesting story:

## Domain-driven design

- One of the purest examples of domain-driven design for a large-scale transactional business application, anywhere in the world
- Extreme re-use and sharing of objects between applications
- Enables easy modification in response to changing business requirements

## Agile Development

- Possibly the first large-scale application of agile development within the public sector, anywhere in the world

## Empowered Users

- A rich user interface to a core transactional business system

## Powerful & Productive Environment

- User interfaces 100% auto-generated from the underlying business objects
  - with no custom coding to write or to maintain
- More opportunity to explore domain than otherwise possible