Welcome to SpringOne

Where Did All My Beautiful Code Go?

Gregor Hohpe  gregor@hohpe.com  www.eaipatterns.com

Google
"We hire only the brightest engineers in the industry."

--Your Company

Still Code Tends To Look Like This...

```java
// This code is wrong. It should use the session manager instead of the parameter manager.
// [joe] I think this is no longer true

/**
 * Returns the customer's account. If there are multiple accounts returns the oldest account.
 * TODO make this less confusing by better supporting the notion of multiple accounts per customer.
 */
Account getOnlyAccount() { ... }

/**
 * Performs a set of arcane checks such as circles etc.
 */
```
No Lack Of Tools & Techniques

- Objects-Oriented Analysis & Design
- Dependency Injection
- Functional Programming
- Aspects
- Light-weight Containers
- Dynamically Typed Languages
- Continuations
- Services
- Workflow Engines
- XML Config File
- Rules Engines
- Objects-Oriented Analysis & Design
- Dependency Injection
- Functional Programming
- Aspects
- Light-weight Containers
- Dynamically Typed Languages
- Continuations
- Services
- Workflow Engines
- XML Config File
- Rules Engines

"No one actually writes the bad code. It magically appears."

--Gregor
1. Understand your domain
2. Choose your model(s)
3. Choose a language to represent the model
4. Map the model well to the language
5. Protect Your Model
6. Program for humans, not machines

A Domain Of Managers And Clients...

- A manager can manage one or more clients
- A manager has to request access to a client's account first ("invite the client")
- A manager can request full access or partial access
- Subsequently, the client can accept or refuse
- A client can later revoke the manager's access
- A client can have a manager for each access type
- Managers can manage other managers, up to 3 levels deep
Choose Your Model(s)

- User Model
  - Domain
  - Structural Model
  - Process Model

Make Request

- View Pending Request
  - Accept Request
  - Confirmation

© 2006 Gregor Hohpe. All rights reserved.
www.eaipatterns.com
Choose A Language

1. A Language whose programming model matches the domain model

2. A general purpose language

- Java
- OSWorkflow
- BPEL

Flows In a Flow Language

```xml
<step id="1" name="Pending">
  <action id="1" name="Accept">
    <results>
      <unconditional-result step="2"/>
    </results>
  </action>
  <action id="2" name="Reject">
    <results>
      <unconditional-result step="3"/>
    </results>
  </action>
</step>
<step id="2" name="Accepted">
  <step id="3" name="Rejected">
```

© 2006 Gregor Hohpe. All rights reserved. www.eaipatterns.com
Programming Flows In An OO Language

```c
seqActivity1 = new SequenceActivity();
seqActivity1.Activities.Add(
    ReceiveRequest);
seqActivity1.Activities.Add(
    ProcessRequest);
seqActivity1.Activities.Add(
    SendConfirmation);
seqActivity1.Name = "sequenceActivity1";
```

Language Trade-Offs

- Switching cost can be high
- Language workbenches try to solve this
- Don’t forget your favorite tools
  - Debugger
  - Refactoring
  - Version Control Integration

© 2006 Gregor Hohpe. All rights reserved.
www.eaipatterns.com
Map The Model Well To The Language

"Invitations expire 30 days after the end of the month in which they were made"

```java
TimeZone zone = TimeZone.getTimeZone("Universal");
Calendar calendar = Calendar.getInstance(zone);
calendar.set(Calendar.YEAR, year);
calendar.set(Calendar.MONTH, month-1);
calendar.set(Calendar.DATE, day);
calendar.set(Calendar.HOUR_OF_DAY, 0);
calendar.set(Calendar.MINUTE, 0);
...
```

Did I say anything about time zone? Hours?

Month - 1 ??

March Is Not A Number

```java
CalendarDate invited = CalendarDate.from(2006, 3, 23);
int allowedDays = 30;
CalendarDate expiryDate =
    completion.month().end().plusDays(allowedDays);
```

- `month()` is a `CalendarInterval` (a range of days, not a number)
- `end()` is a `CalendarDate` (a specific day, not a point in time)
Protect Your Model

- No longer a code issue
- Make it beautiful but accessible
- Anticorruption layer
- Validation tools, e.g. imports etc.
- Good test coverage can actually hurt

Program For Humans Not Machines

- People will read your code
- Make it easy for them
- Misunderstandings are not the user’s but the designer’s fault
- Usability test your interfaces
- Learn from usability design: affordances,…
"Programmers Are People, Too"

- Kevlin Henney
  - "Effective Interface Design"
  - Affordances (Don Norman's Design of Everyday Things)

- Josh Block
  - "How to Design a Good API and Why it Matters"
  - "Conceptual Weight" of an API

- Ken Arnold
  - "Programmers are people, Too"

"Write code worth reading"

-- Ward Cunningham
Thank You!