Sneaking Scala

Into the Enterprise

@davetron5000 / dave @ opower.com

A (Java Developer's) Tour of Scala - naildrivin5.com/scalatour

naildrivin5.com/blog

Slides on Github github.com/davetron5000/sneaking-scala

Slides online sneaking-scala.heroku.com

Sneaking Scala

Into the Enterprise

Why?

Barriers

Solutions

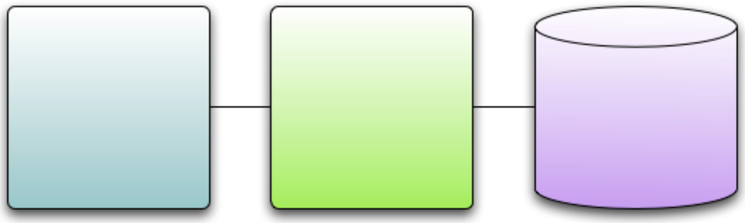
Engineering Lead @

OPOWER
ENERGY EFFICIENCY. DELIVERED

15 years professional developer

10+ years of Java

Built many apps like this:



OPOWER
ENERGY EFFICIENCY. DELIVERED

(Where I work)

Software as a Service

Java Shop

We actually Make Money

We are married only to what works

How could Scala *work* for your company?

or, Why do we care in the first place?

Get more done

More expressive

Fewer lines of code

More productive

Fewer bugs?

Talent attractor

Natural progression on the JVM

Natural progression on the JVM

What's in the way?

Steep learning curve

New Syntax

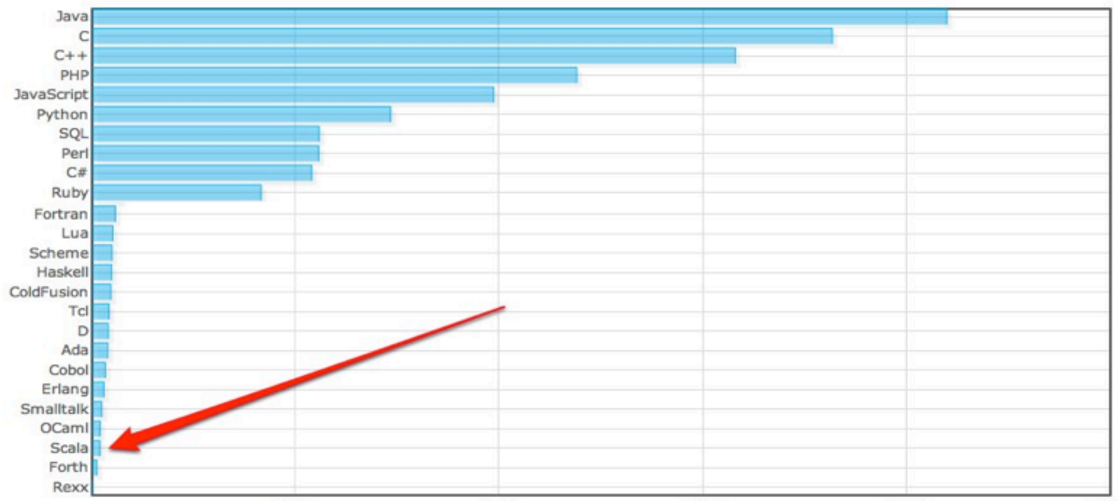
Many new concepts

Docs, books, community all in early stages

Where do I even *find* a Scala developer?

Normalized Comparison

This is a chart showing combined results from all data sets.



http://langpop.com/

Java's delivering

How do we make this happen?

Fight the learning curve

Sneak it in

Control Risk

History Lesson C++ to Java

Replace existing apps/write new ones

New libraries

New deployment mechanism

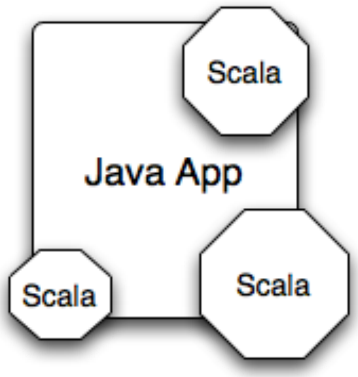
Java to Scala

Replace *components*

Reuse libraries

Same deployment mechanism

Java to Scala



Don't need to abandon Java

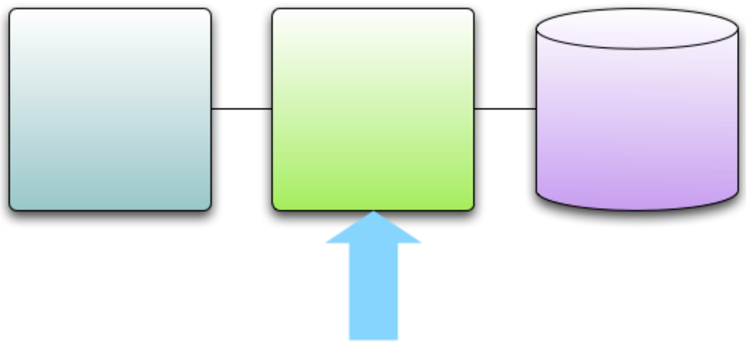
Don't need the entire Scala language (or library) to get started

Getting started carries less risk

Is this realistic?

And does this actually provide value?

Business Logic



Fully Armed and Operational Programming Language

http://www.flickr.com/photos/flying_cloud/2667225198

Hard to control Learning Curve

```
class PersonService {
  this: PersonDAO with UtilityDAO =>

  def login(name:String, password:String) = {
    if (checkPassword(name,password))
      val token = recordLogin(name)
      Some(token)
    else
      None
  }
}
```

Hard to control Learning Curve

```
class PersonService {
  this: PersonDAO with UtilityDAO =>
//^^^^^ What is this even called?!^^^
  def login(name:String, password:String) = {
    if (checkPassword(name,password))
      val token = recordLogin(name)
      Some(token)
    else
      None
  }
}
```

High Risk

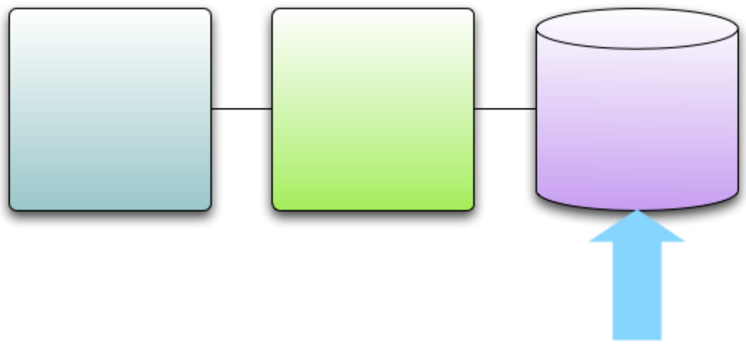
(though arguably high value)

A win in the long term

Productivity, Maintainability, Learnability hits in the short term

Requires on-site or in-house experts (expensive)

Model/Persistence Layer



Model Objects

```
public class Person {
    private String firstName;
    private String lastName;
    private Date birthdate;
    private char gender;
    private String email;
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public Date getBirthdate() {
        return birthdate;
    }
    public void setBirthdate(Date birthdate) {
        this.birthdate = birthdate;
    }
    public char getGender() {
```

Model Objects

```
class Person(
  var firstName:String,
  var lastName:String,
  var birthdate:Date,
  var gender:Char,
  var email:String)
```

Model Objects

```
case class Person(
  @BeanProperty var firstName:String,
  @BeanProperty var lastName:String,
  @BeanProperty var birthdate:Date,
  @BeanProperty var gender:Char,
  @BeanProperty var email:String)
```

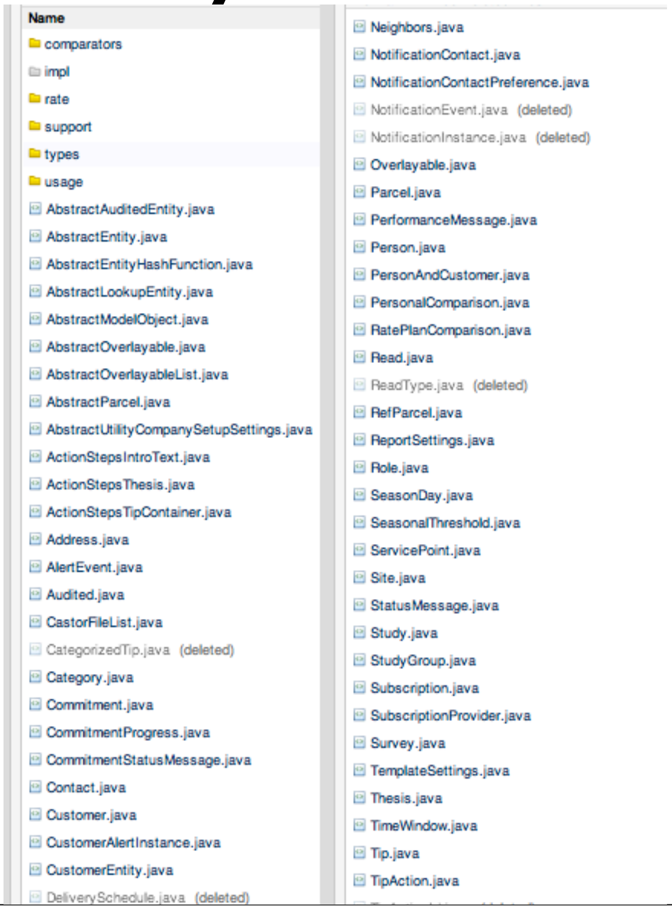
Low Risk, High Value

Very little Scala Knowledge

HUGE reduction in code size

Two Problems

Already have a ton

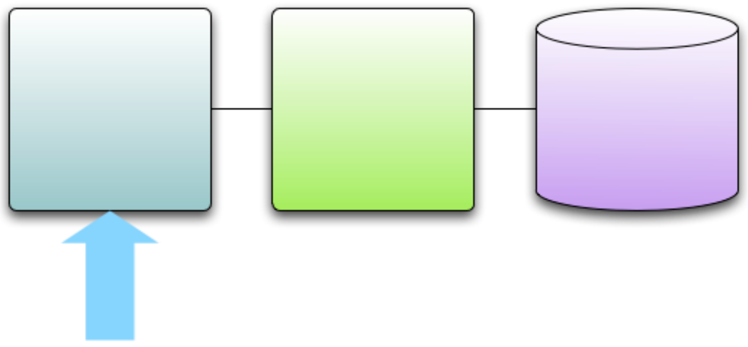


```
Name
comparators
impl
rate
support
types
usage
AbstractAuditedEntity.java
AbstractEntity.java
AbstractEntityHashFunction.java
AbstractLookupEntity.java
AbstractModelObject.java
AbstractOverlayable.java
AbstractOverlayableList.java
AbstractParcel.java
AbstractUtilityCompanySetupSettings.java
ActionStepsIntroText.java
ActionStepsThesis.java
ActionStepsTipContainer.java
Address.java
AlertEvent.java
Audited.java
CastorFileList.java
CategorizedTip.java  (deleted)
Category.java
Commitment.java
CommitmentProgress.java
CommitmentStatusMessage.java
Contact.java
Customer.java
CustomerAlertInstance.java
CustomerEntity.java
DeliverySchedule.java  (deleted)
```

```
Neighbors.java
NotificationContact.java
NotificationContactPreference.java
NotificationEvent.java  (deleted)
NotificationInstance.java  (deleted)
Overlayable.java
Parcel.java
PerformanceMessage.java
Person.java
PersonAndCustomer.java
PersonalComparison.java
RatePlanComparison.java
Read.java
ReadType.java  (deleted)
RefParcel.java
ReportSettings.java
Role.java
SeasonDay.java
SeasonalThreshold.java
ServicePoint.java
Site.java
StatusMessage.java
Study.java
StudyGroup.java
Subscription.java
SubscriptionProvider.java
Survey.java
TemplateSettings.java
Thesis.java
TimeWindow.java
Tip.java
TipAction.java
```

Not a significant source of bugs

What about application endpoints?



DaveWeb5000 Controller

```
public class TipController {

  public Object getTip() {
    String id = params.get("id");
    String format = params.get("format");
    Tip tip = tipService.find(id);
    return formatAs(format,tip);
  }
}
```

Scala-ized version

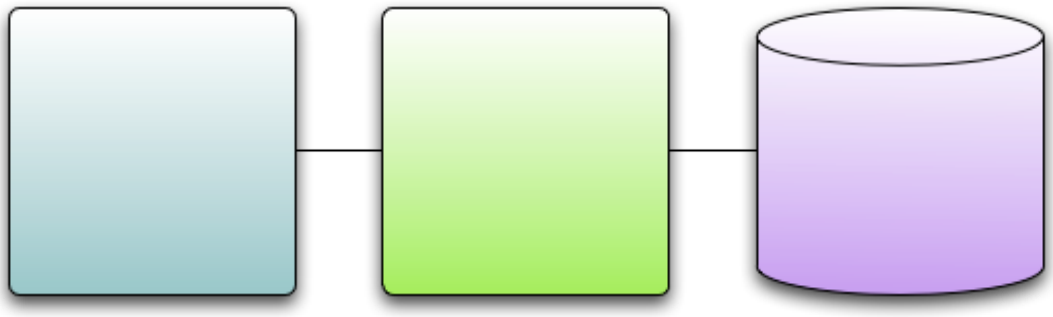
```
class TipController {

  def getTip = {
    val id = params("id");
    val format = params("format");
    val tip = tipService.find(id);
    formatAs(format,tip);
  }
}
```

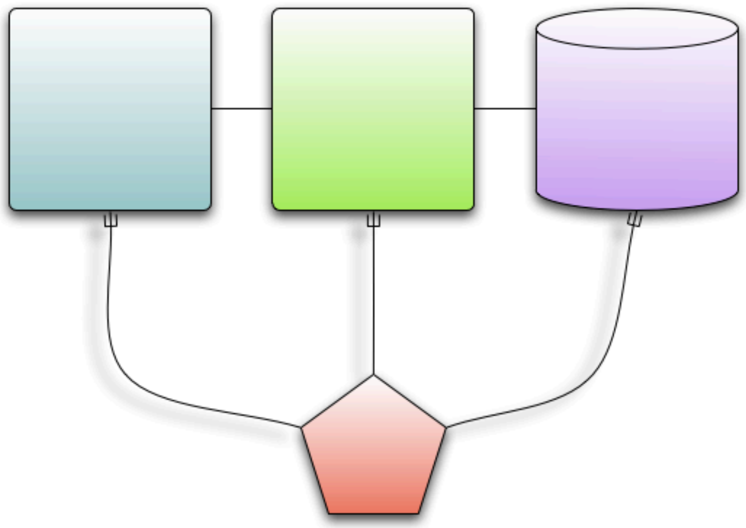
"Power of Scala" has little to add

Low Risk, Low Value

All is lost?



Testing!



No new deployment dependencies

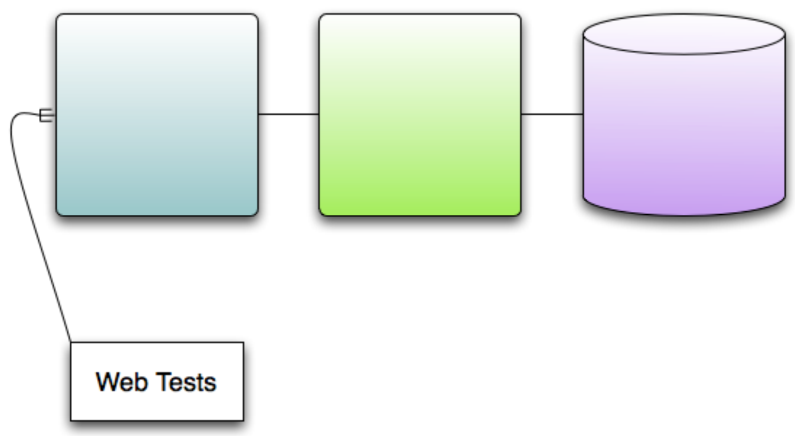
Can focus on your problem domain

Tests have a lot of boilerplate

Scala makes boilerplate go away

What OPOWER did

Web Testing



Web Testing

New Concept for team

Key to success of product

New API to learn for everyone

Can we sneak in Scala?

(and add value :)

HTMLUnit/JWebUnit is like assembly language

```
public void testLogin {
  tester = new WebTester();
  tester.gotoURL(HOME_PAGE);
  tester.gotoURL(PROTECTED_PAGE);
  tester.assertOnForm("login");
  tester.setValue("user","dave@blah.com");
  tester.setValue("pass","foobar69");
  tester.submit();
  tester.assertOn(PROTECTED_PAGE);
  // Real tests much longer and
  // more painful
}
```

DSL for web testing *our* app

```
class WebTestLogin extends WebTestSpec {
  page("protected",
       (page:PageSpecification) => {

    page.requiresLogin()
    page.shouldContain(
      "Hello Dave").inElement("h1")
  })
}
```

Smooth the learning curve

Ground Rules for DSL

Consistent syntax

No new operators

Minimize new concepts

How many new concepts?

```
class WebTestLogin extends WebTestSpec {
  page("protected", // *1*
       (page:PageSpecification) => { // *2,3*

    page.requiresLogin()
    page.shouldContain(
      "Hello Dave").inElement("h1")
  })
}
```

[1] Basic Syntax - constructor code

```
class WebTestLogin extends WebTestSpec {
  page("protected", // *1* ...
```

[2] types come after a :

```
class WebTestLogin extends WebTestSpec {
  page("protected", // *1*
       (page:PageSpecification) => { // *2*
```

[3] Anonymous functions/closures

```
class WebTestLogin extends WebTestSpec {
  page("protected",
       (page:PageSpecification) => { // *3*
        ^^^^param                  ^^^^function
```

Reinforces familiar concepts

Dots - x.y == method call - easy to understand

page - has an obvious type, we can look up its methods

Ground Rules for Implementation

Had a weekend to build it

Mixins, Case Classes, Collections All fair game

Imperative/OO design

Ground Rules for Implementation

Avoid (initially) confusing features

No implicits

Minimize type parameters

Ground Rules for Implementation

Tutorial, scaffolds, documentation

Lots of scaladoc

"How" and "What" Comments

Couldn't this have been done in Java?

Certainly, but...

DSL implementation would've taken too long in Java

Conclusions

Scala *can* provide value to your organization

A gradual introduction minimizes risk, maximizes value

Testing is an easy win

Tame the learning curve

Small successes socialize its value

Gradually expand

Thank You

@davetron5000 / dave @ opower.com

Slides on Github github.com/davetron5000/sneaking-scala

Slides online sneaking-scala.heroku.com