OW TO PICK A FUT
OF JAVA ARCHITE
THE NEXT 10+ Y

ko Roić

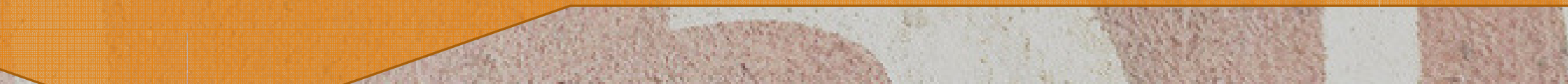
visor @
ng ICT

Matija Capar

Chief architect
King ICT

o live in the present often wi
he present to an extent that i
he possibility of having a futu

UCLA/MIT/Atari/Turing award



OBLEM? THINGS CHA
IT HARD TO STAY ON
0+ YEARS. EVEN IN A
ED EVOLVING
CTURES. EVEN IF YOU
AS POSSIBLE"

easons for even

ultiple angles and

long term Java


NEEDS
GROWING
SCALABILITY
FOR THE ENTIR
LIFESPAN
NEEDS
FUNCTIONAL
UPGRADES FOR
THE ENTIRE

nt

ge

Client operating systems change

Application runtimes change (e.

Systems you integrated to also

JVMs change

Server infrastructure changes

Security standards change

nt

ge

Young guns know only the laste

Seniors are hard to find

...especially for non standard arc

...especially for middleware-hea

People (and knowledge) leave t

nt

ge

Allowing functional growth over

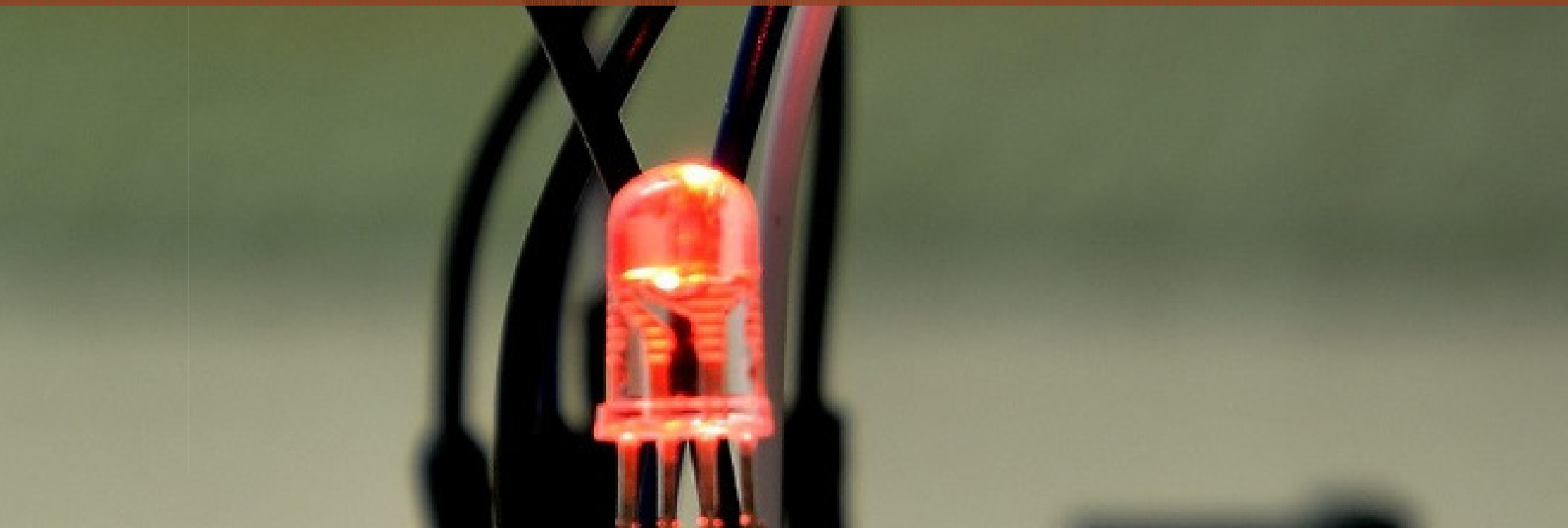
Allowing capacity growth over t

Allowing easy deployment and

Allowing partial or full refactorin


LUTION, PLEASE? J
CH TODAY, BUILD YO
SOLUTION
olution

ARE GOING TO DO
OM UP SO PLEASE
WITH US ☺



architecture is the collection of
system's quality attributes; w
s and are hardest to change.
provides the frame within wh
e) is built."

m-Gal-Oz in Agile Architectu

UNTIME. JVM? WHI
VERSION?

AVA 7 WAS MADE AV
Y 2011. AND MADE D

GUAGE. THERE'S M
HAN 50 ON THE JVI

THE NEW COBOL. IT WILL STICK LI
U PICKED UP ON A BACHELOR'S PA

NDEX, JOB MARKET INDEX, COLLE

N TARGET FOR THE NE

YEARS?

T: *THERE IS THE EE AND T*

ACH. EACH WILL LIVE TO

ABOUT THE IMPLICATION

OVERALL SOLUTION?

VELOP REALLY THAT

Y USE FRAMEWORKS MODULES.

ERE'S THOUSANDS OF FRA

DULES AROUND, WE'LL TR

ME GUIDANCE ON PICKING

SUSTAINABLE ONES

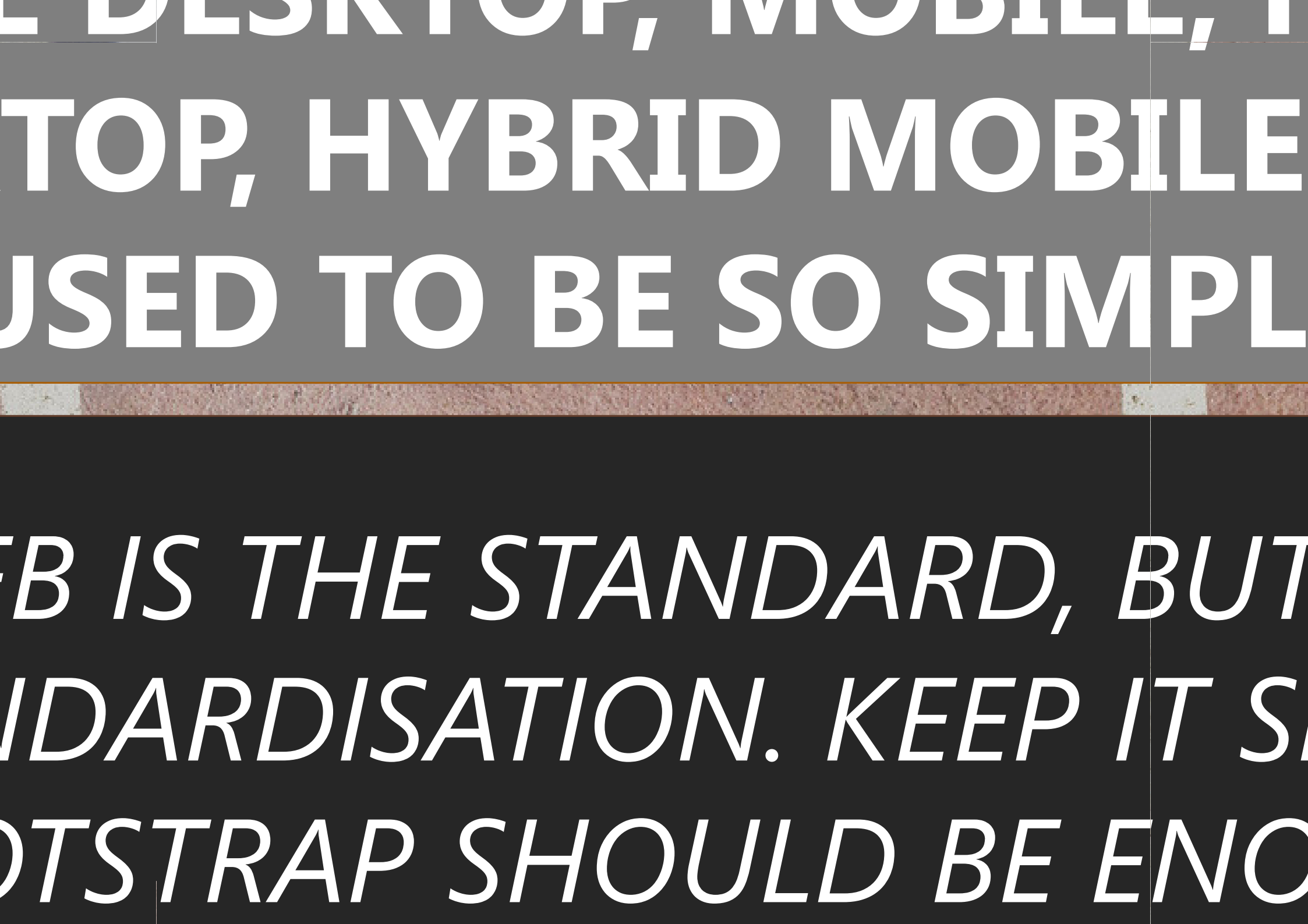
RECORD OF BACKWARD COMPATIBLITY

ENGAGEM

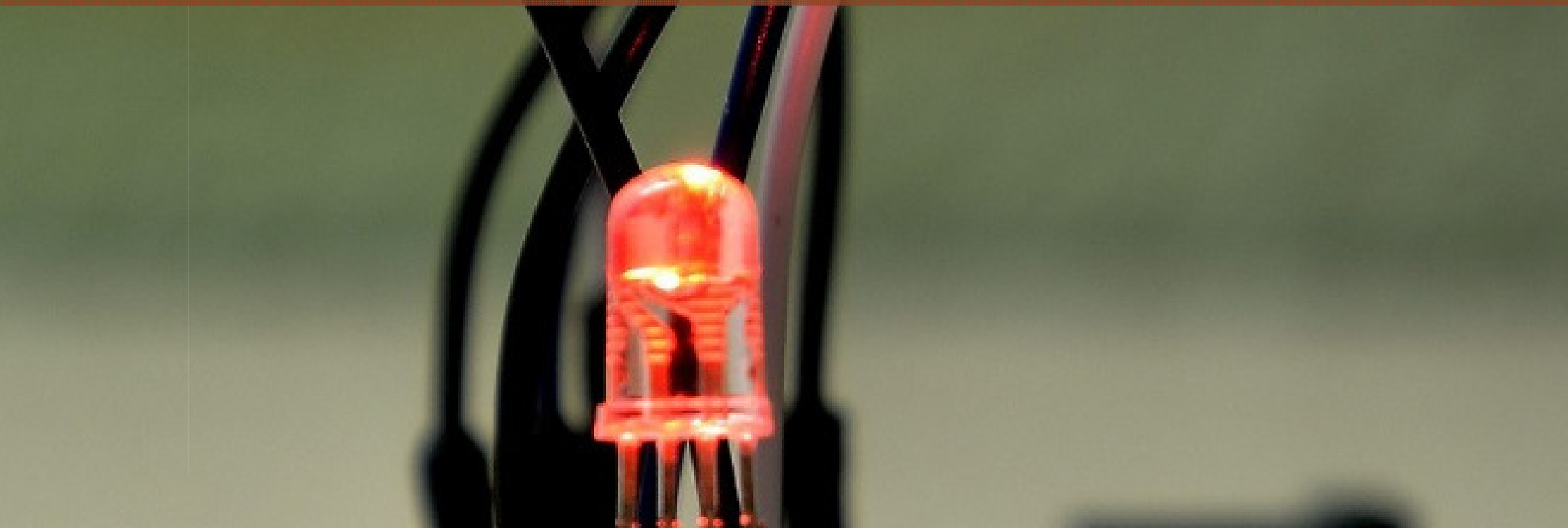
DO

NE

IS IT EASY FOR A JUNIOR TO


E DESKTOP, MOBILE, T
TOP, HYBRID MOBILE
USED TO BE SO SIMPL
EB IS THE STANDARD, BUT
NDARDISATION. KEEP IT SI
OTSTRAP SHOULD BE ENO

FTWARE ARCHITEC
LAND. IT SHARES S
TS SYSTEM ARCHIT



ADVANCED MIDDLEWARE

Architecture based on BPM or similar

Can you afford the knowledge transfer?

Can you afford scaling the server up?

DIRECT INTEGRATION	*ESB APPROACH*
Smaller operating cost	Higher operating
Difficult to manage	Scalability built
Difficult to upgrade	Future proof

FINAL WORDS..

MAINTAINING KNOWLEDGE

Use Java as the default language

Use as few frameworks as possible

Do not build your own frameworks, unless you are Google

D ONE MORE THI


TIVE!