REST
Roy Fielding, Dissertation 2000

- Roy Fielding

- *Chief Scientist*, Day Software
- *Co-founder and member*, The Apache Software Foundation
- *Dissertation on Architectural Styles and the Design of Network-based Software Architectures at the* Information and Computer Science, UC Irvine

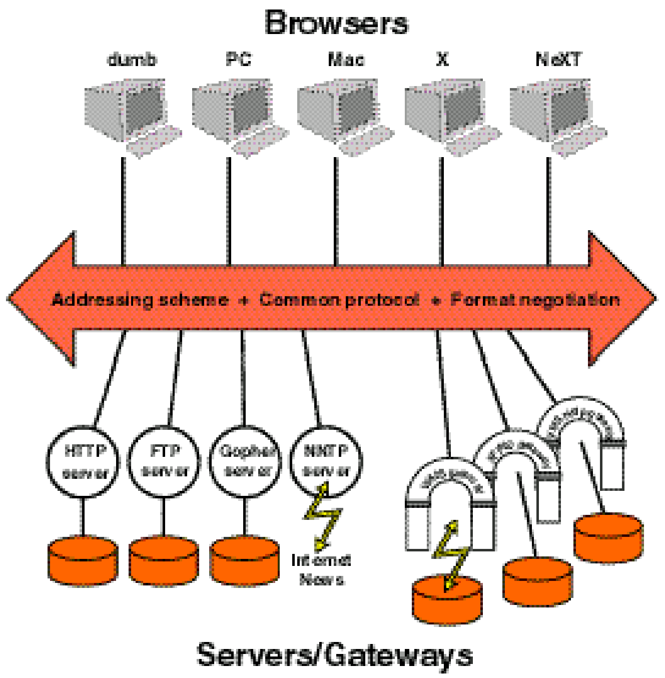
In his dissertation, he „*introduce[s] the Representational State Transfer (REST) architectural style and describe[s] how REST has been used to guide the design and development of the architecture for the modern Web*“

Ressources:

- Chapter 5, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- http://roy.gbiv.com/talks/200709_fielding_rest.pdf

Has played a role in authoring the Internet standards for the Hypertext Transfer Protocol (HTTP) and Uniform Resource Identifiers (URI)

The early web

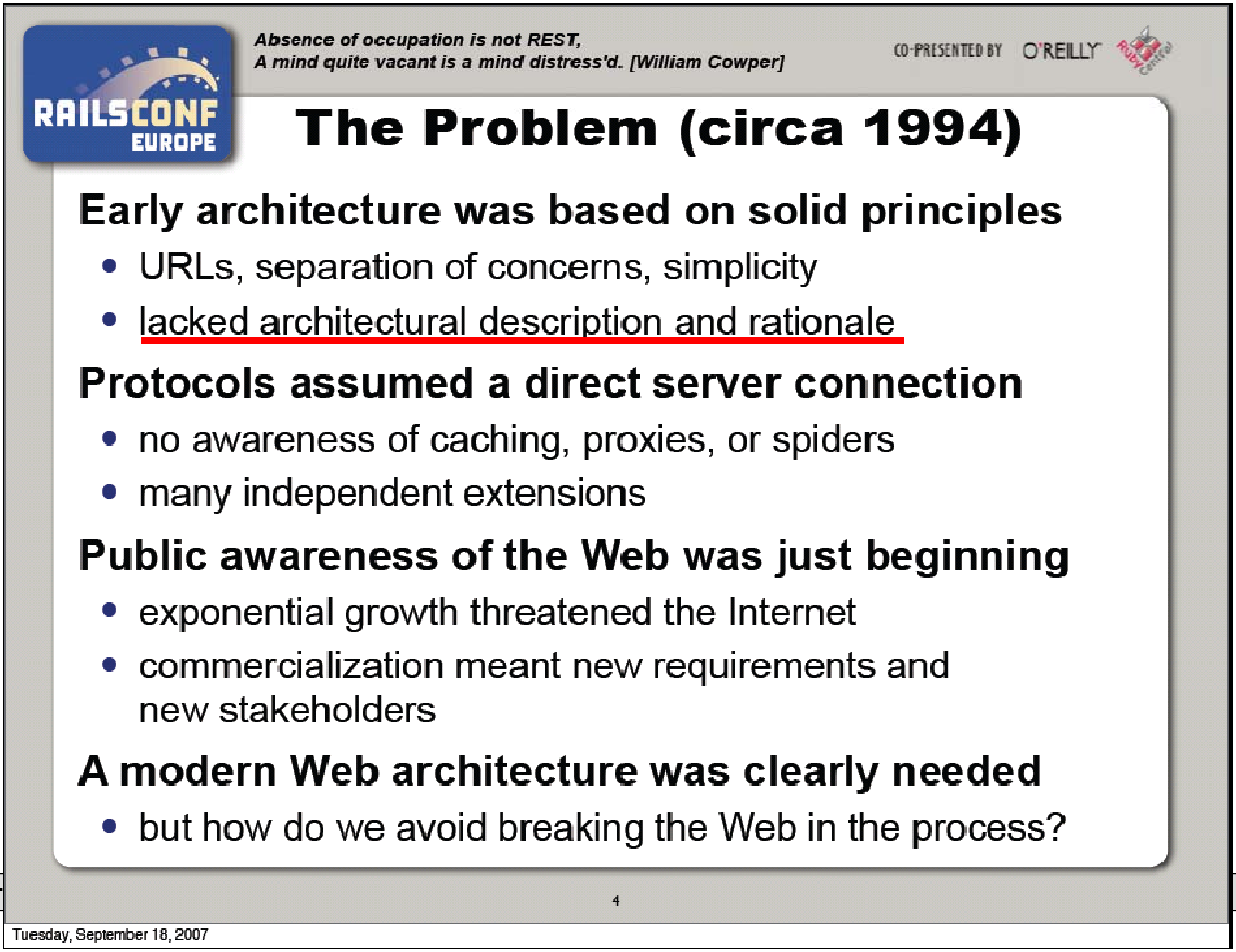


© 1992 Tim Berners-Lee, Robert Cailliau, Jean-François Groff, C.E.R.N.

Figure 5-5. Early WWW Architecture Diagram

REST

Roy Fielding, RailsConf Europe, September 2007



Absence of occupation is not REST,
A mind quite vacant is a mind distress'd. [William Cowper]

CO-PRESENTED BY O'REILLY

The Problem (circa 1994)

Early architecture was based on solid principles

- URLs, separation of concerns, simplicity
- lacked architectural description and rationale

Protocols assumed a direct server connection

- no awareness of caching, proxies, or spiders
- many independent extensions

Public awareness of the Web was just beginning

- exponential growth threatened the Internet
- commercialization meant new requirements and new stakeholders

A modern Web architecture was clearly needed

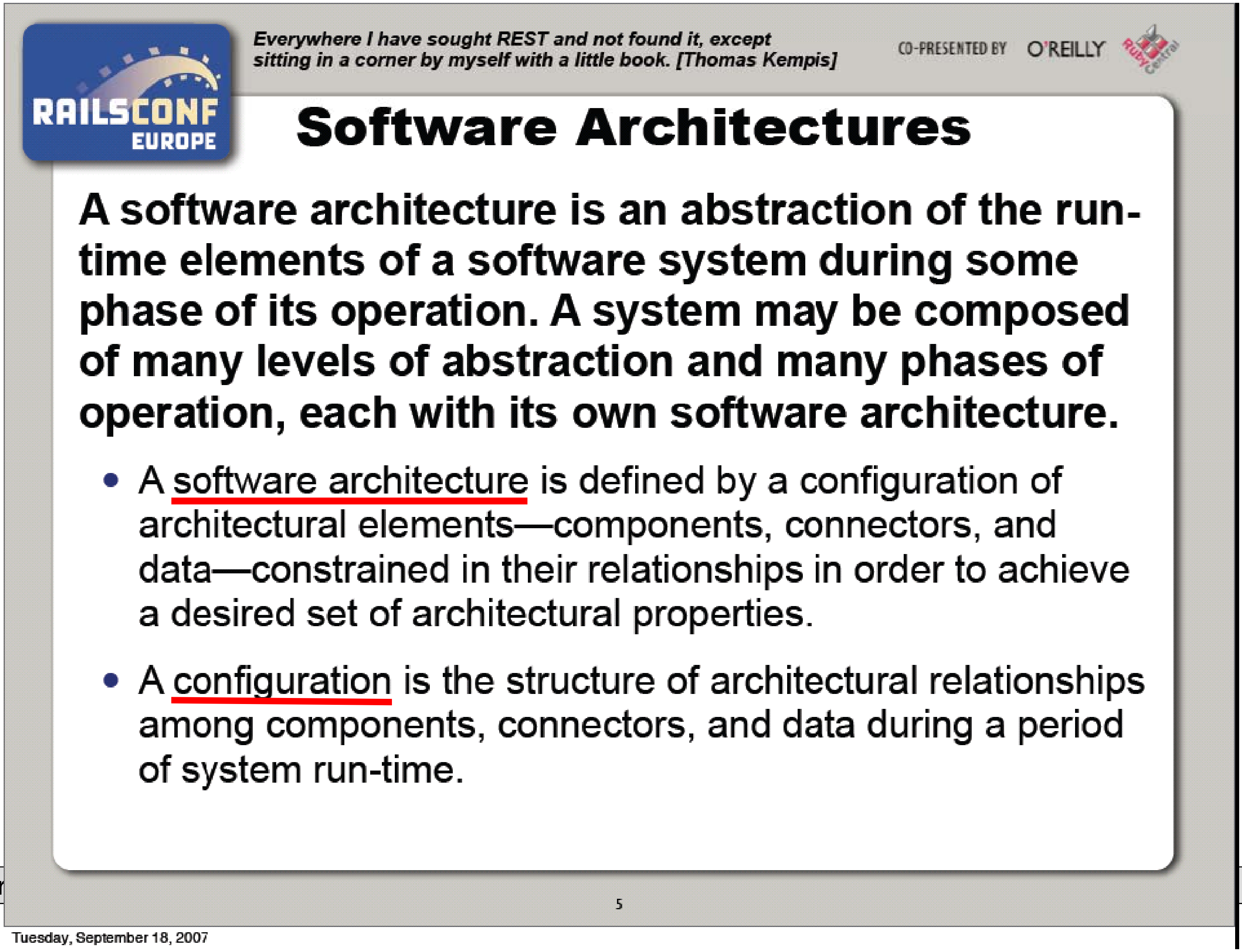
- but how do we avoid breaking the Web in the process?

4

Tuesday, September 18, 2007

REST

Roy Fielding, RailsConf Europe, September 2007



RAILSCONF EUROPE

Everywhere I have sought REST and not found it, except sitting in a corner by myself with a little book. [Thomas Kempis]

CO-PRESENTED BY O'REILLY Ruby Central

Software Architectures

A software architecture is an abstraction of the run-time elements of a software system during some phase of its operation. A system may be composed of many levels of abstraction and many phases of operation, each with its own software architecture.

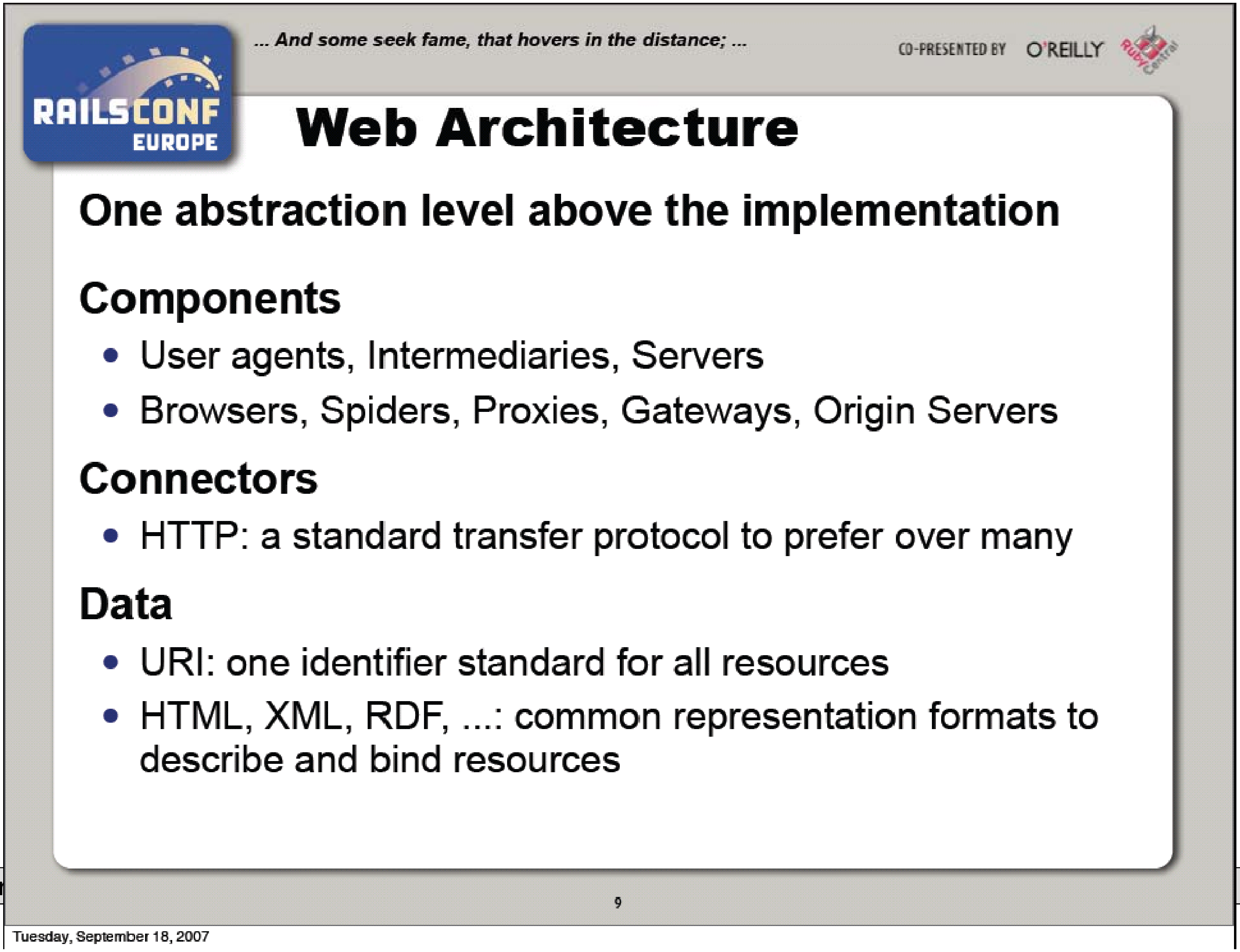
- A software architecture is defined by a configuration of architectural elements—components, connectors, and data—constrained in their relationships in order to achieve a desired set of architectural properties.
- A configuration is the structure of architectural relationships among components, connectors, and data during a period of system run-time.

5

Tuesday, September 18, 2007

REST

Roy Fielding, RailsConf Europe, September 2007



... And some seek fame, that hovers in the distance; ...

CO-PRESENTED BY O'REILLY

Web Architecture

One abstraction level above the implementation

Components

- User agents, Intermediaries, Servers
- Browsers, Spiders, Proxies, Gateways, Origin Servers

Connectors

- HTTP: a standard transfer protocol to prefer over many

Data

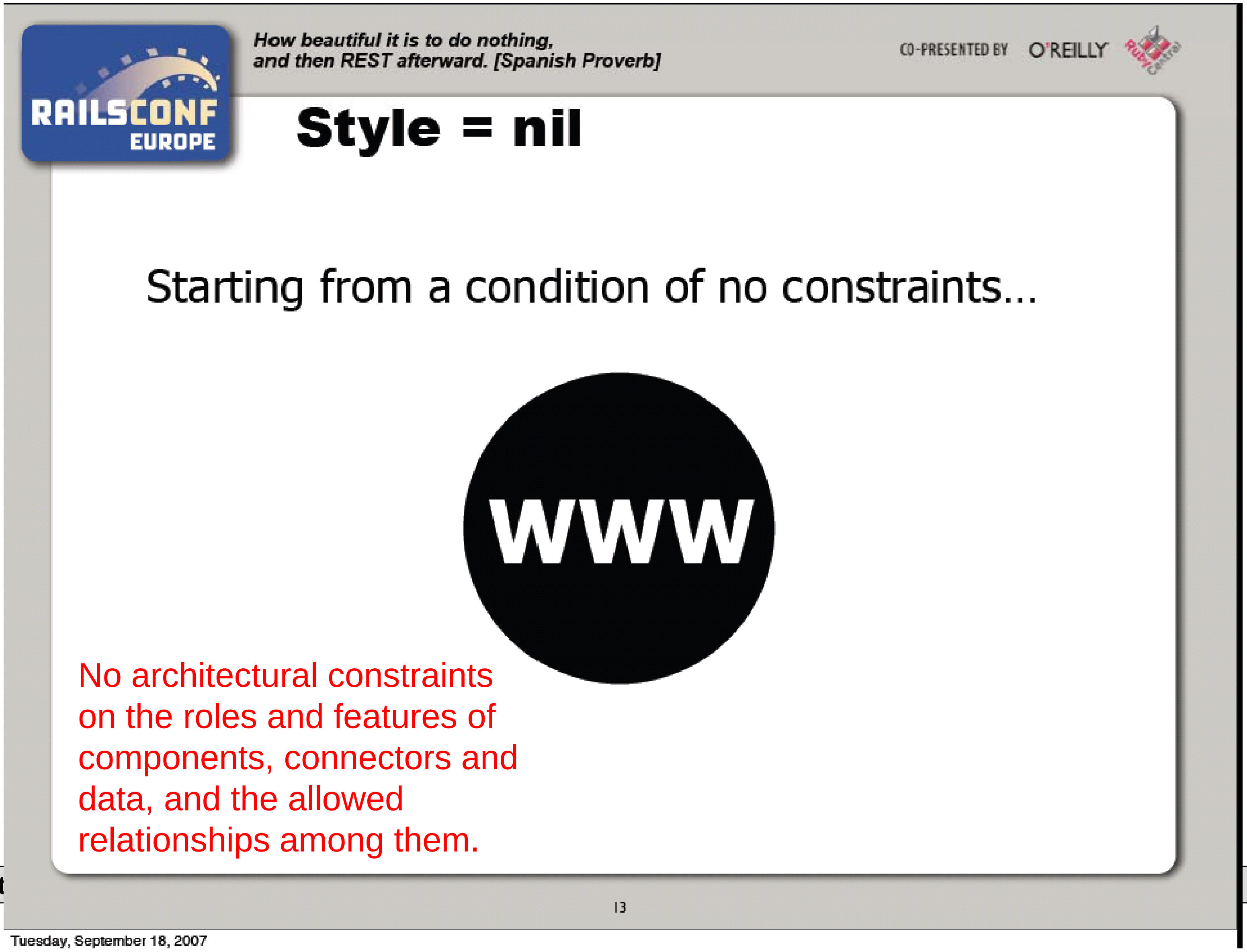
- URI: one identifier standard for all resources
- HTML, XML, RDF, ...: common representation formats to describe and bind resources

9

Tuesday, September 18, 2007

REST

Roy Fielding, RailsConf Europe, September 2007



RAILSCONF EUROPE

How beautiful it is to do nothing, and then REST afterward. [Spanish Proverb]

CO-PRESENTED BY O'REILLY

Style = nil

Starting from a condition of no constraints…

No architectural constraints on the roles and features of components, connectors and data, and the allowed relationships among them.

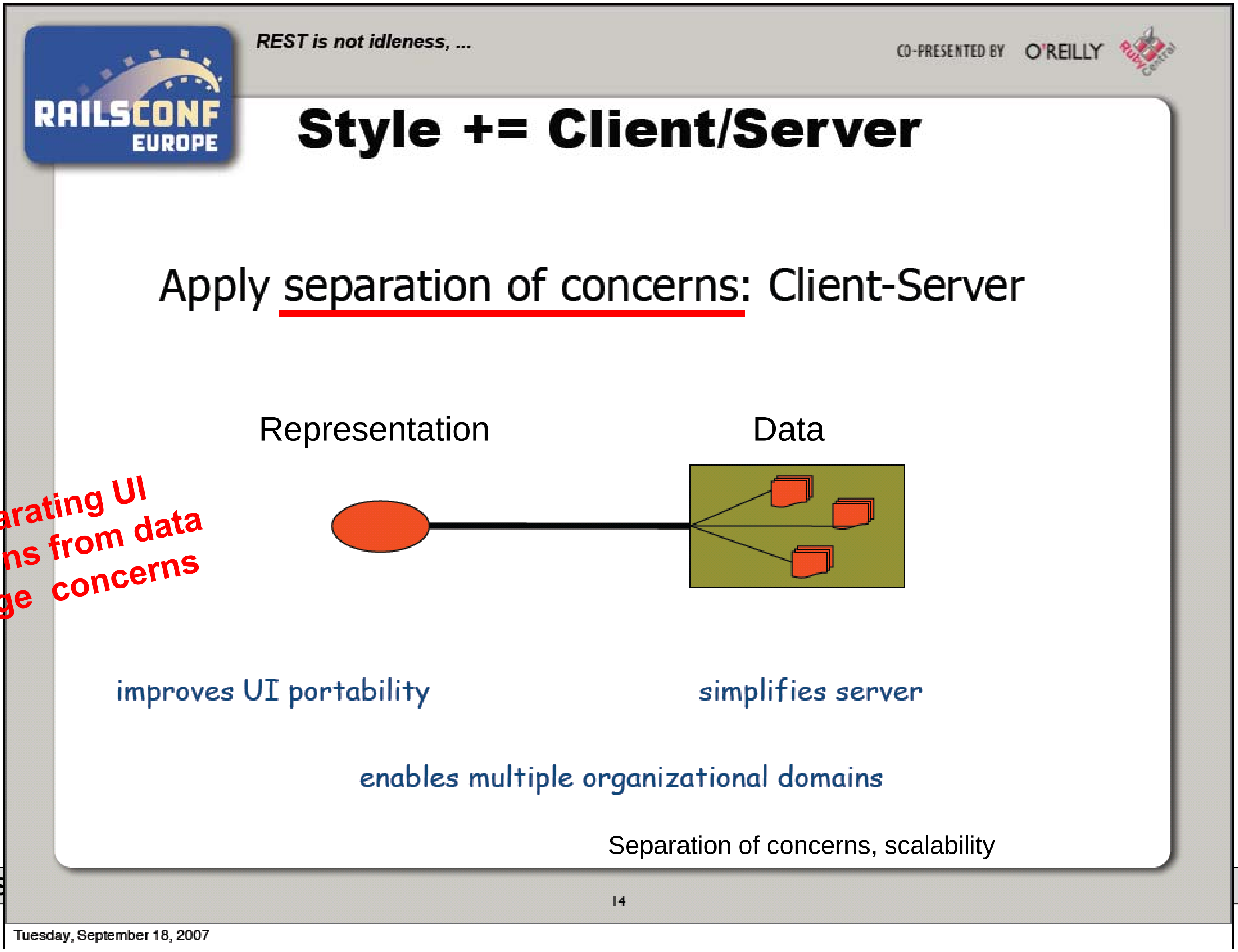
13

Tuesday, September 18, 2007

REST

client-server (CSS) style

Roy Fielding, RailsConf Europe, September 2007



REST is not idleness, ...

CO-PRESENTED BY O'REILLY

Style += Client/Server

Apply separation of concerns: Client-Server

improves UI portability

simplifies server

enables multiple organizational domains

Separation of concerns, scalability

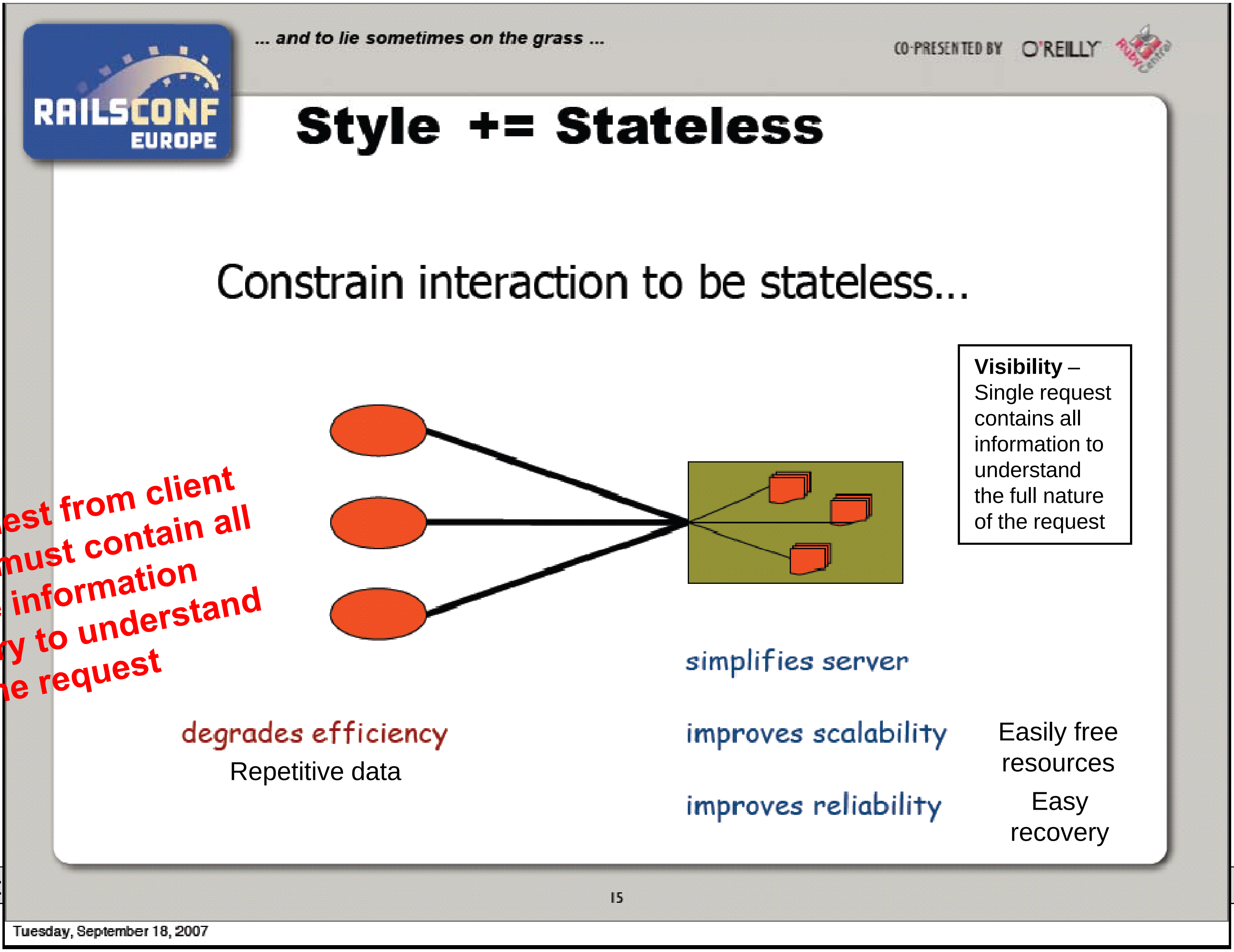
14

Tuesday, September 18, 2007

Separating UI concerns from data storage concerns

REST client-stateless-server (CSS) style

Roy Fielding, RailsConf Europe, September 2007



... and to lie sometimes on the grass ...

CO-PRESENTED BY O'REILLY

Style += Stateless

Constrain interaction to be stateless...

Visibility – Single request contains all information to understand the full nature of the request

each request from client to server must contain all of the information necessary to understand the request

simplifies server

degrades efficiency

Repetitive data

improves scalability

Easily free resources

improves reliability

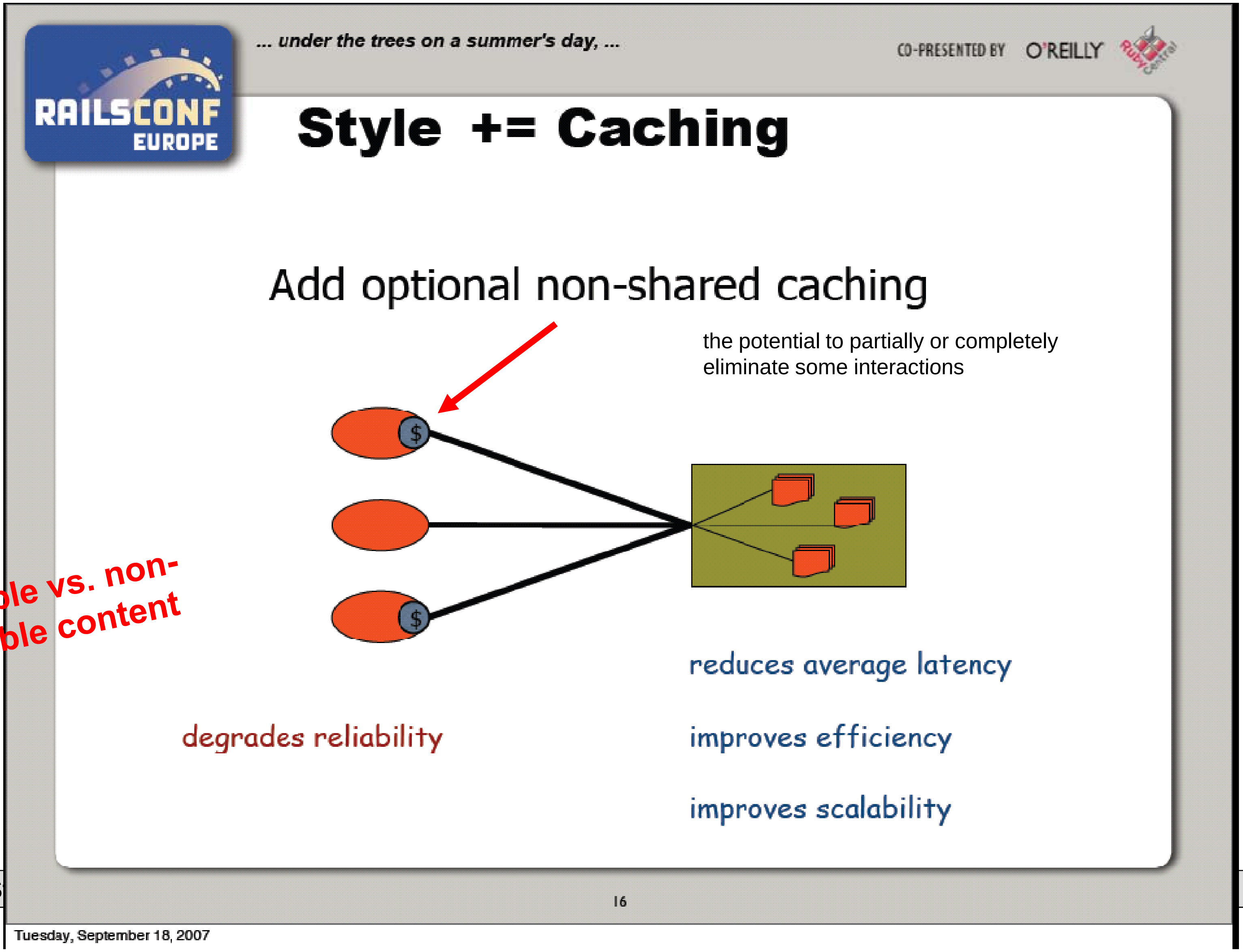
Easy recovery

15

Tuesday, September 18, 2007

REST client-cache-stateless-server style

Roy Fielding, RailsConf Europe, September 2007



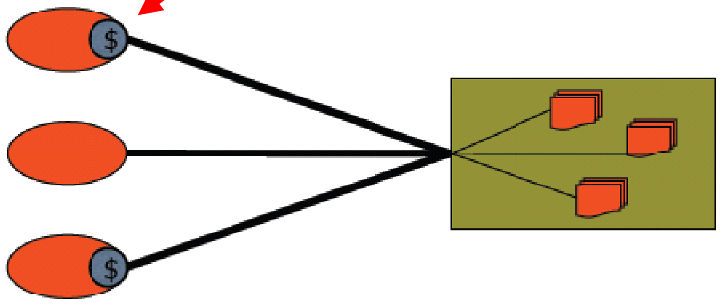
... under the trees on a summer's day, ...

CO-PRESENTED BY O'REILLY

Style += Caching

Add optional non-shared caching

the potential to partially or completely eliminate some interactions



Cachable vs. non-cachable content

degrades reliability

reduces average latency

improves efficiency

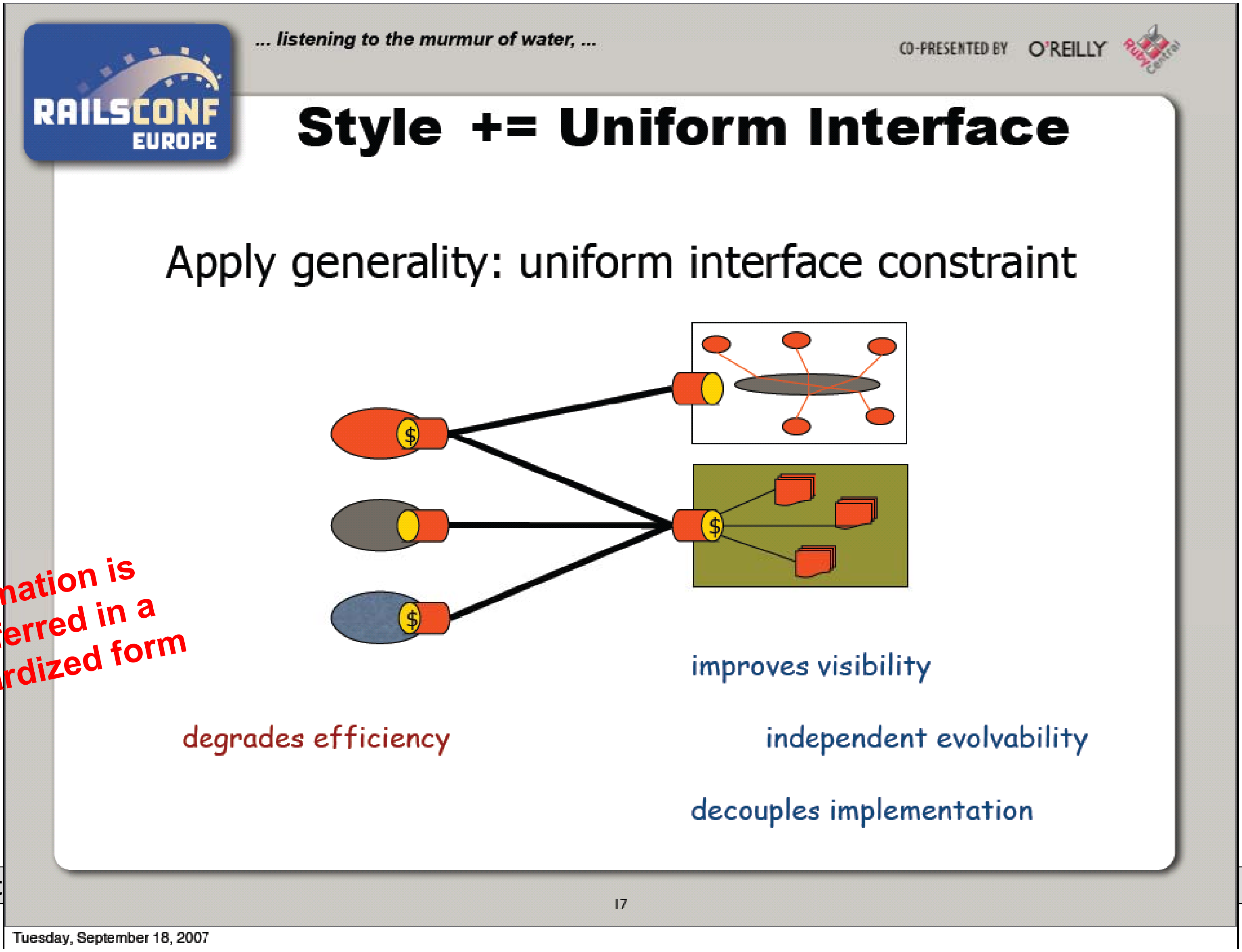
improves scalability

16

Tuesday, September 18, 2007

REST

Roy Fielding, RailsConf Europe, September 2007



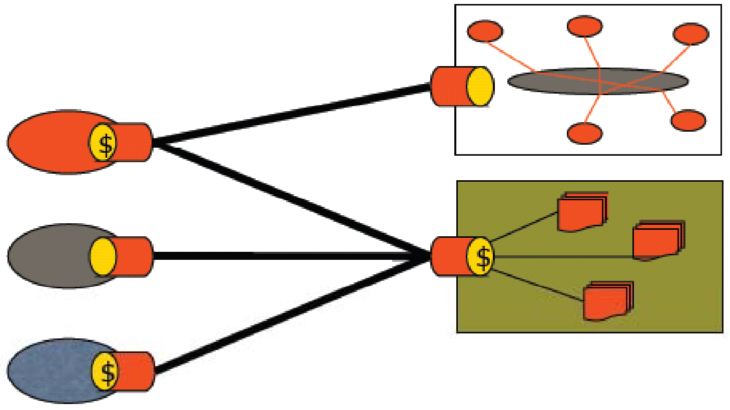
... listening to the murmur of water, ...

RAILSCONF EUROPE

CO-PRESENTED BY O'REILLY RubyCentral

Style += Uniform Interface

Apply generality: uniform interface constraint



improves visibility

degrades efficiency

independent evolvability

decouples implementation

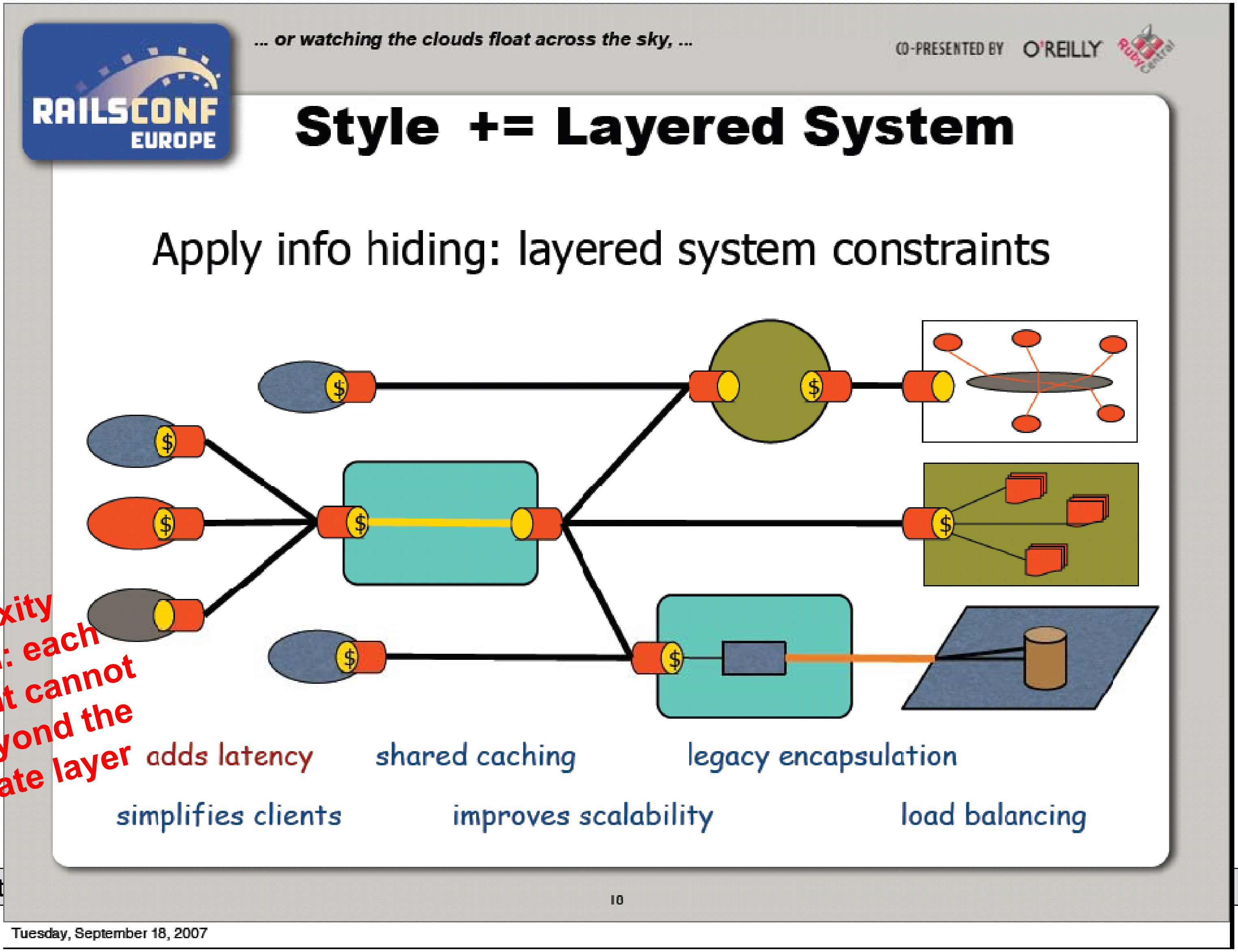
17

Tuesday, September 18, 2007

information is transferred in a standardized form

REST

Roy Fielding, RailsConf Europe, September 2007



RAILSCONF EUROPE

... or watching the clouds float across the sky, ...

CO-PRESENTED BY O'REILLY Ruby Central

Style += Layered System

Apply info hiding: layered system constraints

adds latency

shared caching

legacy encapsulation

simplifies clients

improves scalability

load balancing

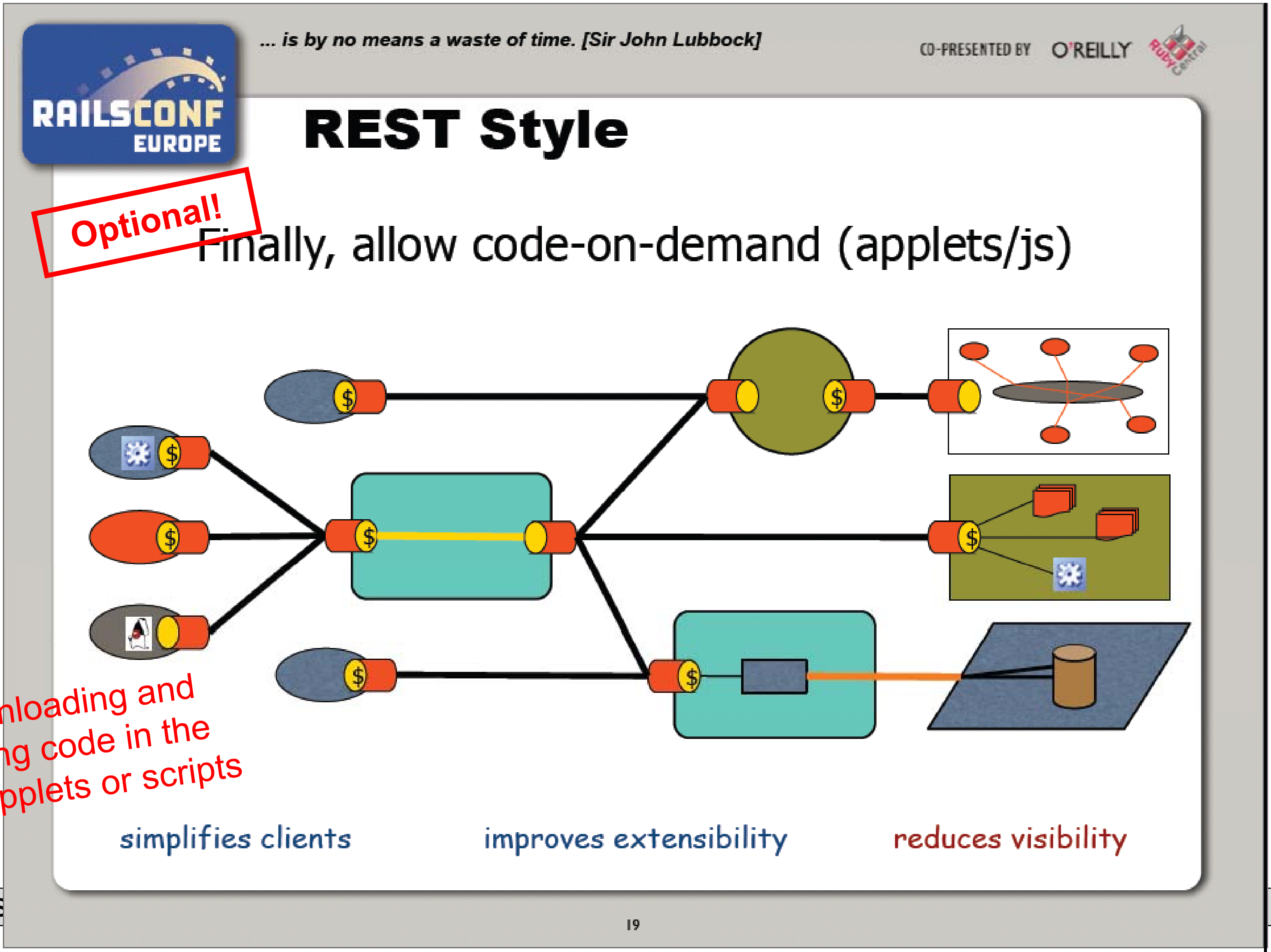
10

Tuesday, September 18, 2007

Complexity reduction: each component cannot "see" beyond the immediate layer

REST

Roy Fielding, RailsConf Europe, September 2007



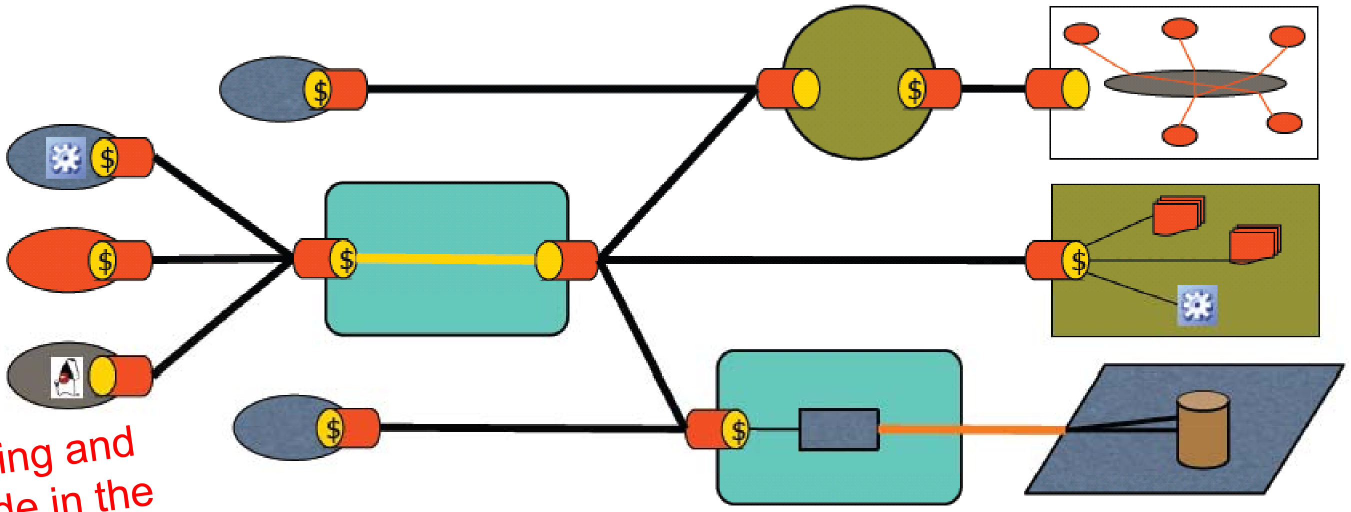
... is by no means a waste of time. [Sir John Lubbock]

CO-PRESENTED BY O'REILLY

REST Style

Optional!

Finally, allow code-on-demand (applets/js)



simplifies clients improves extensibility reduces visibility

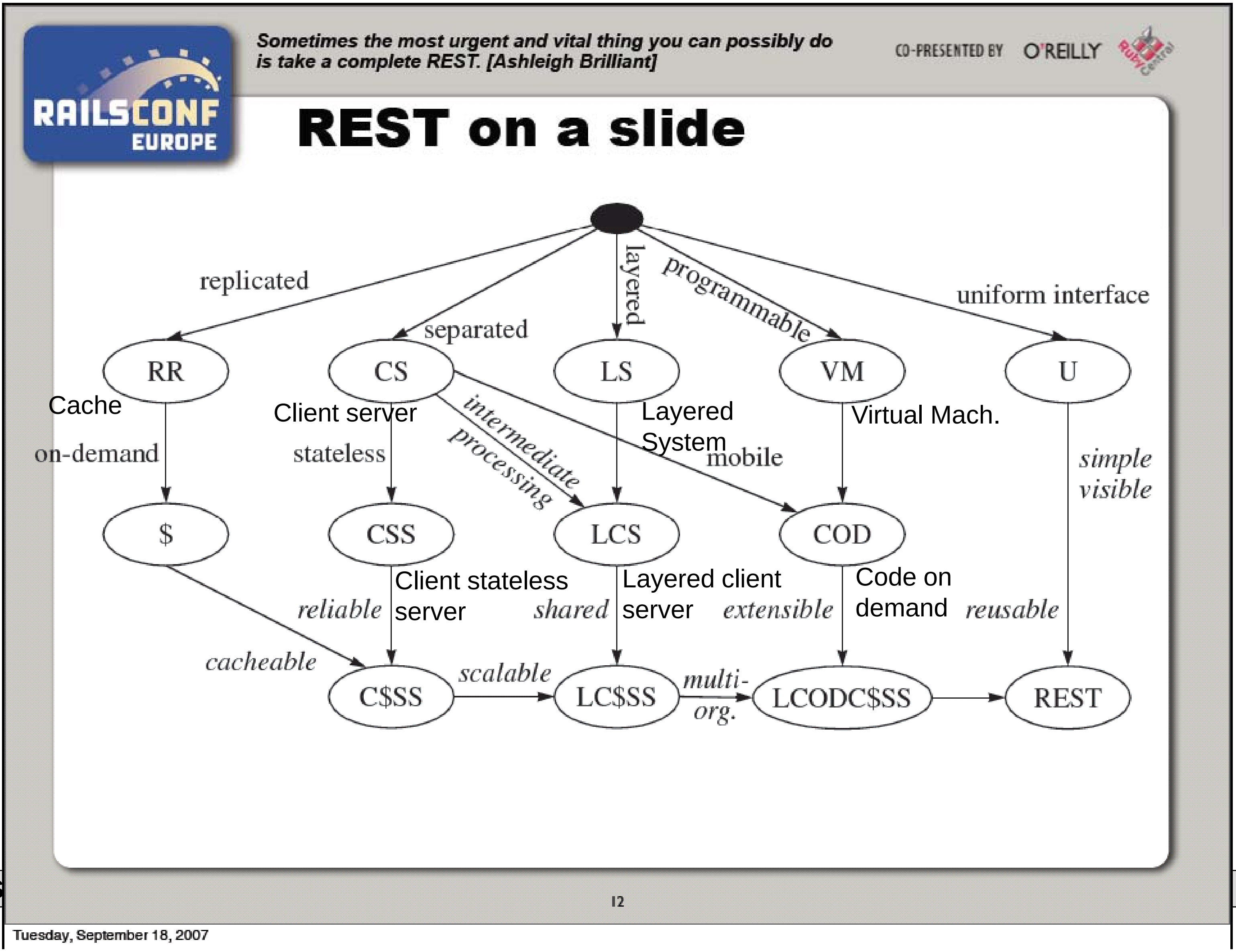
19

Tuesday, September 18, 2007

by downloading and executing code in the form of applets or scripts

REST

Roy Fielding, RailsConf Europe, September 2007



Sometimes the most urgent and vital thing you can possibly do is take a complete REST. [Ashleigh Brilliant]

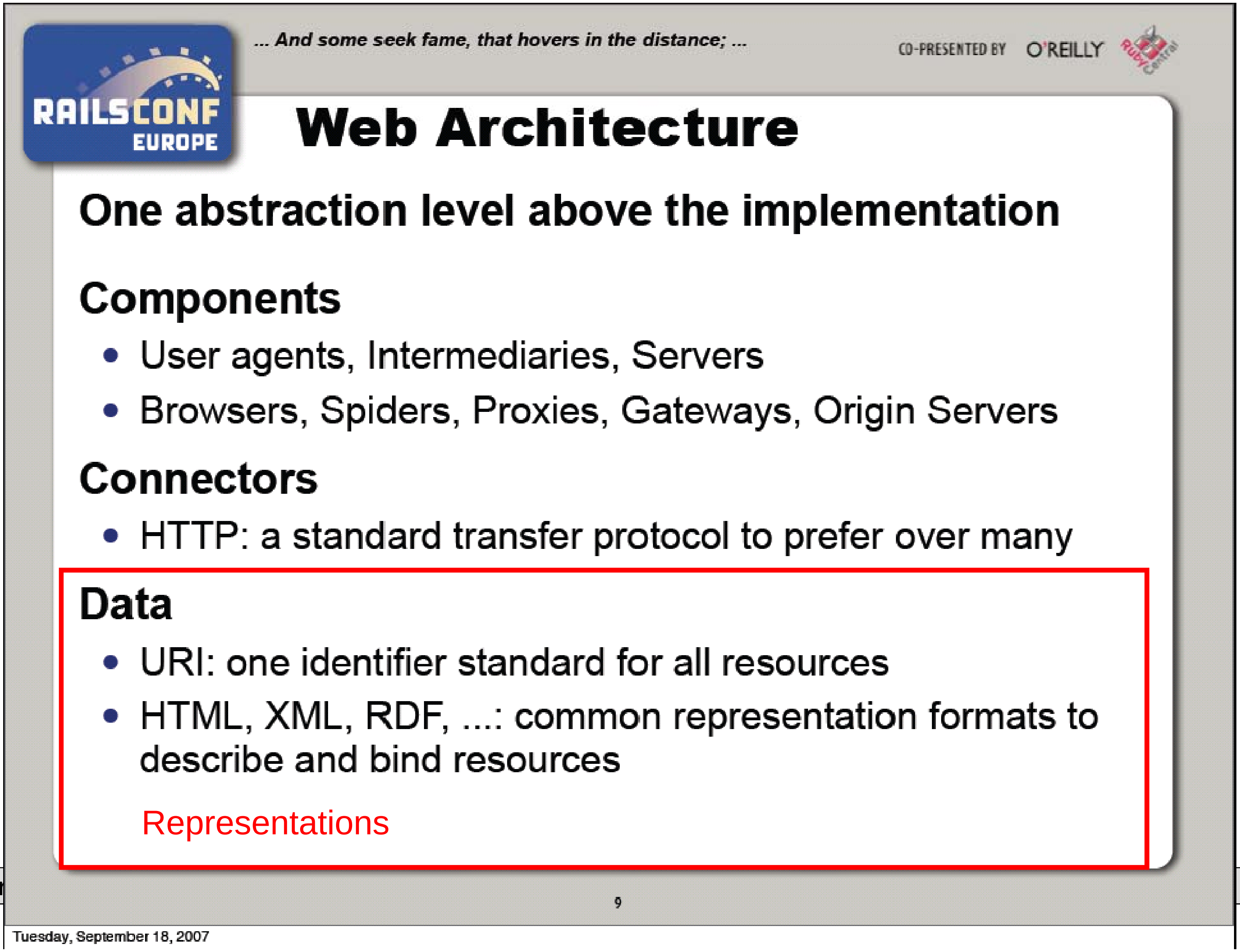
CO-PRESENTED BY O'REILLY

REST on a slide

Tuesday, September 18, 2007

REMINDER: Web Architecture

Roy Fielding, RailsConf Europe, September 2007



RAILSCONF EUROPE

... And some seek fame, that hovers in the distance; ...

CO-PRESENTED BY O'REILLY

Web Architecture

One abstraction level above the implementation

Components

- User agents, Intermediaries, Servers
- Browsers, Spiders, Proxies, Gateways, Origin Servers

Connectors

- HTTP: a standard transfer protocol to prefer over many

Data

- URI: one identifier standard for all resources
- HTML, XML, RDF, ...: common representation formats to describe and bind resources

Representations

9

Tuesday, September 18, 2007

Representations

Roy Fielding, Dissertation 2000

- *“REST components perform* ***actions*** *on a* ***resource*** *by using a* ***representation*** *to capture the* ***current*** *or* ***intended*** *state of that resource and transferring that representation between* ***components****.”*

- *“less precise names for a representation include: document, file, and HTTP message entity, instance, or variant. “*

- *Depending on the message control data, a given representation may indicate the current state of the requested resource, the desired state for the requested resource, or the value of some other resource […].*

REST

Chapter "Representational State Transfer (REST)" in "Pro PHP XML and Web Services", R. Richards 633--672 (2006)

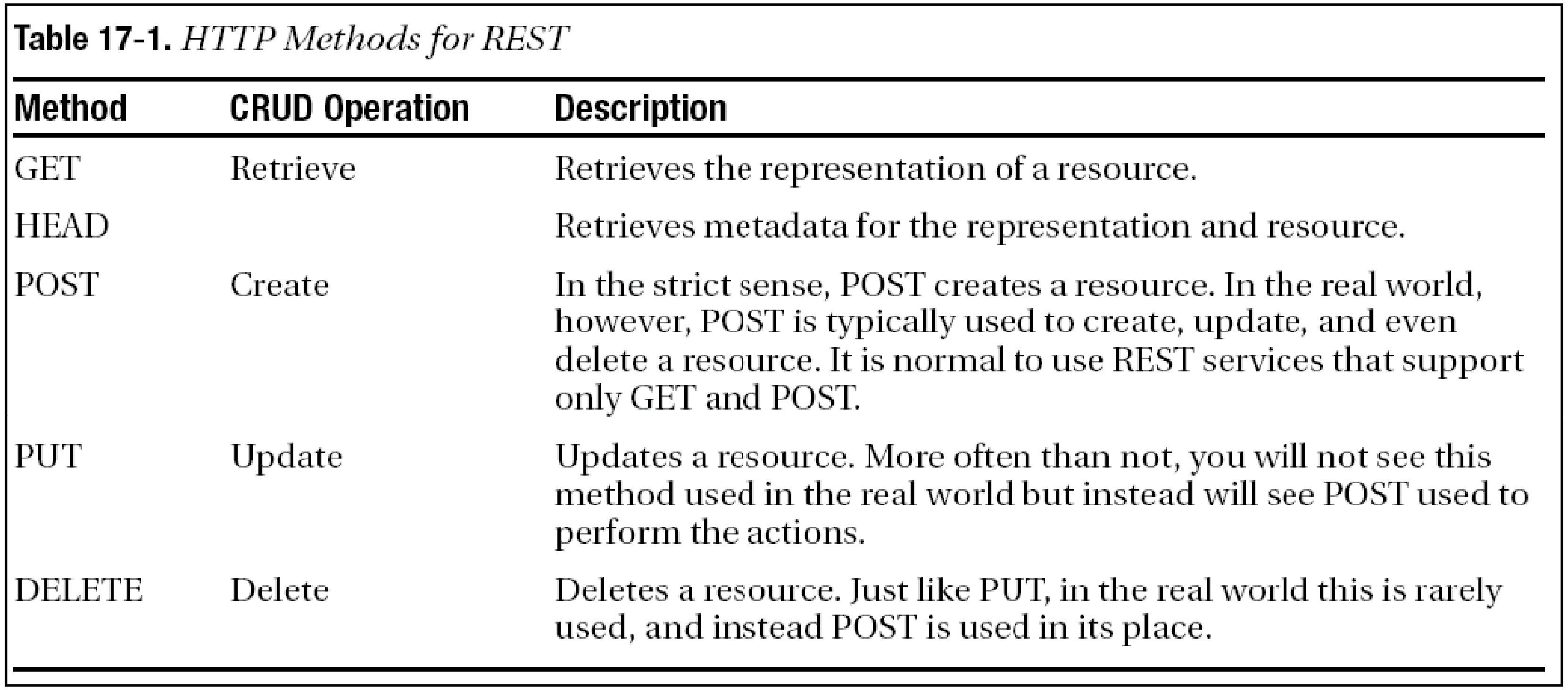


Table 17-1. *HTTP Methods for REST*

Method	CRUD Operation	Description
GET	Retrieve	Retrieves the representation of a resource.
HEAD		Retrieves metadata for the representation and resource.
POST	Create	In the strict sense, POST creates a resource. In the real world, however, POST is typically used to create, update, and even delete a resource. It is normal to use REST services that support only GET and POST.
PUT	Update	Updates a resource. More often than not, you will not see this method used in the real world but instead will see POST used to perform the actions.
DELETE	Delete	Deletes a resource. Just like PUT, in the real world this is rarely used, and instead POST is used in its place.

Theory vs. Practice

Chapter "Representational State Transfer (REST)" in "Pro PHP XML and Web Services", R. Richards 633--672 (2006) and

- How has this theory influenced current practice?

REST applied to HTTP

• The REST service is expressed **as a URL** and is accessed with **basic HTTP requests**,

• The **HTTP verb** is important: a GET is a read operation, POST is a creation, and PUT make updates to the service.

• The return payload is usually **XML** or **JSON**.

http://dev2dev.bea.com/pub/a/2007/05/google-mashups.html