

# Architecture of Enterprise Applications 1

## Overview of Enterprise Applications

2021/9/16 Review

Haopeng Chen

**RE**liable, **IN**telligent and **S**calable Systems Group (**REINS**)

Shanghai Jiao Tong University

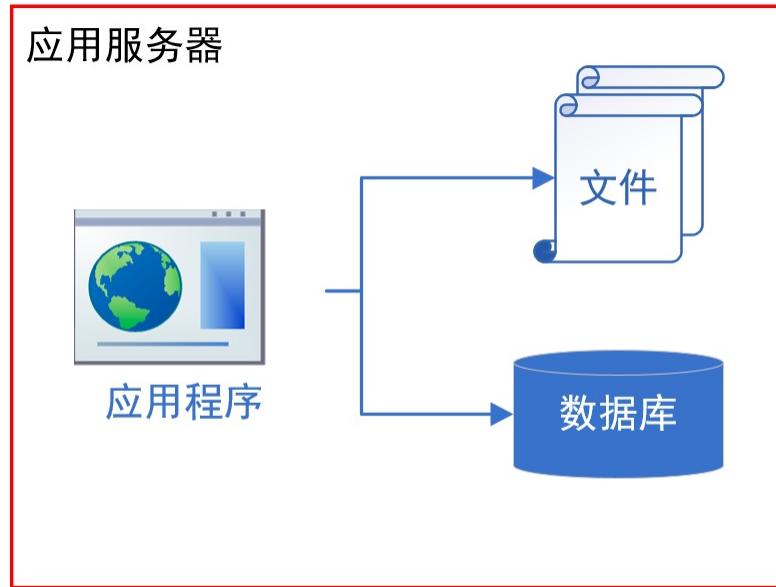
Shanghai, China

<http://reins.se.sjtu.edu.cn/~chenhp>

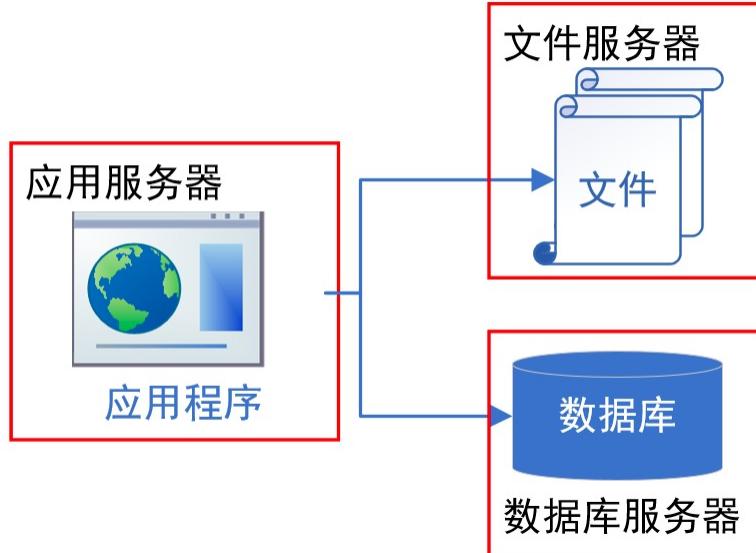
e-mail: chen-hp@sjtu.edu.cn

- **Contents**
  - Architecture
  - Scope
- **Objectives**
  - 能够根据应用系统的规模，确定适合的技术路线，从单机部署到分布式集群部署，再到微服务架构和云部署
  - 能够根据系统需求，设计并实现由合理的有状态服务与无状态服务构成服务层

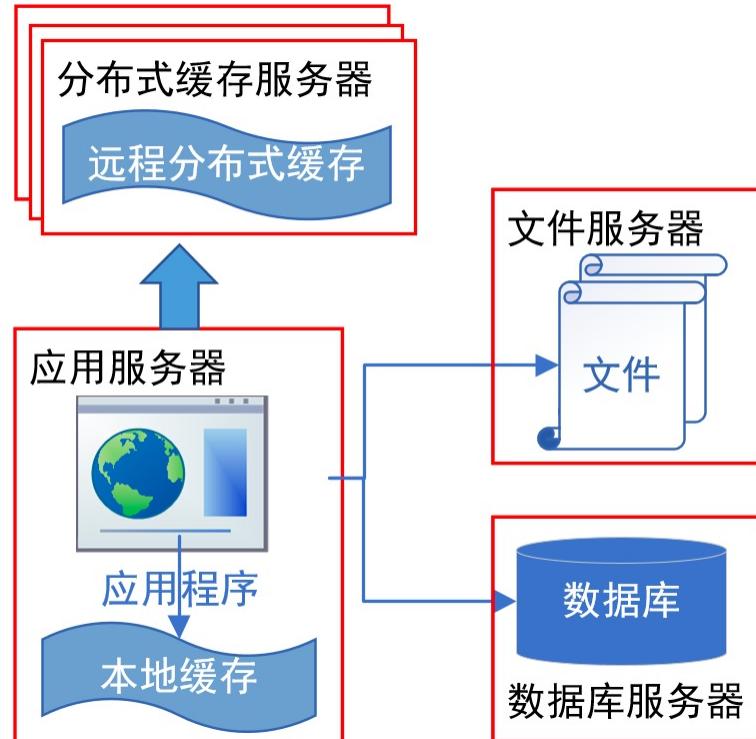
# Architecture of Java Web Application



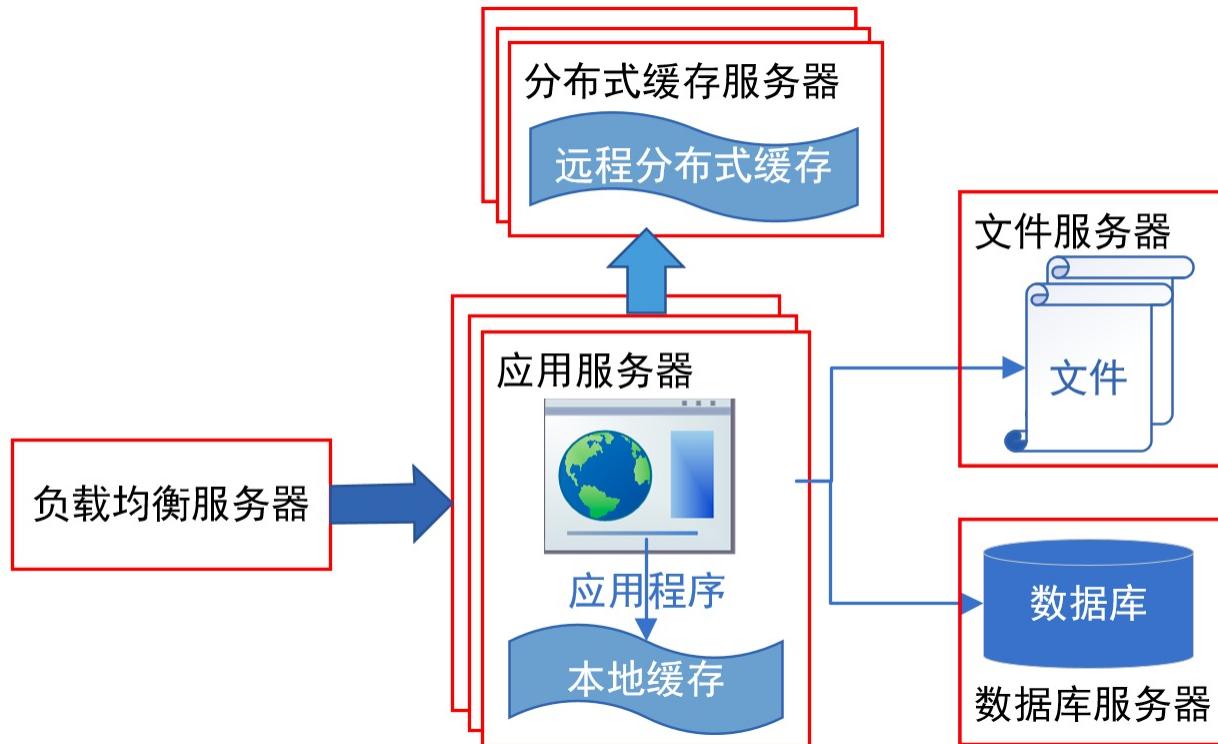
# Architecture of Java Web Application



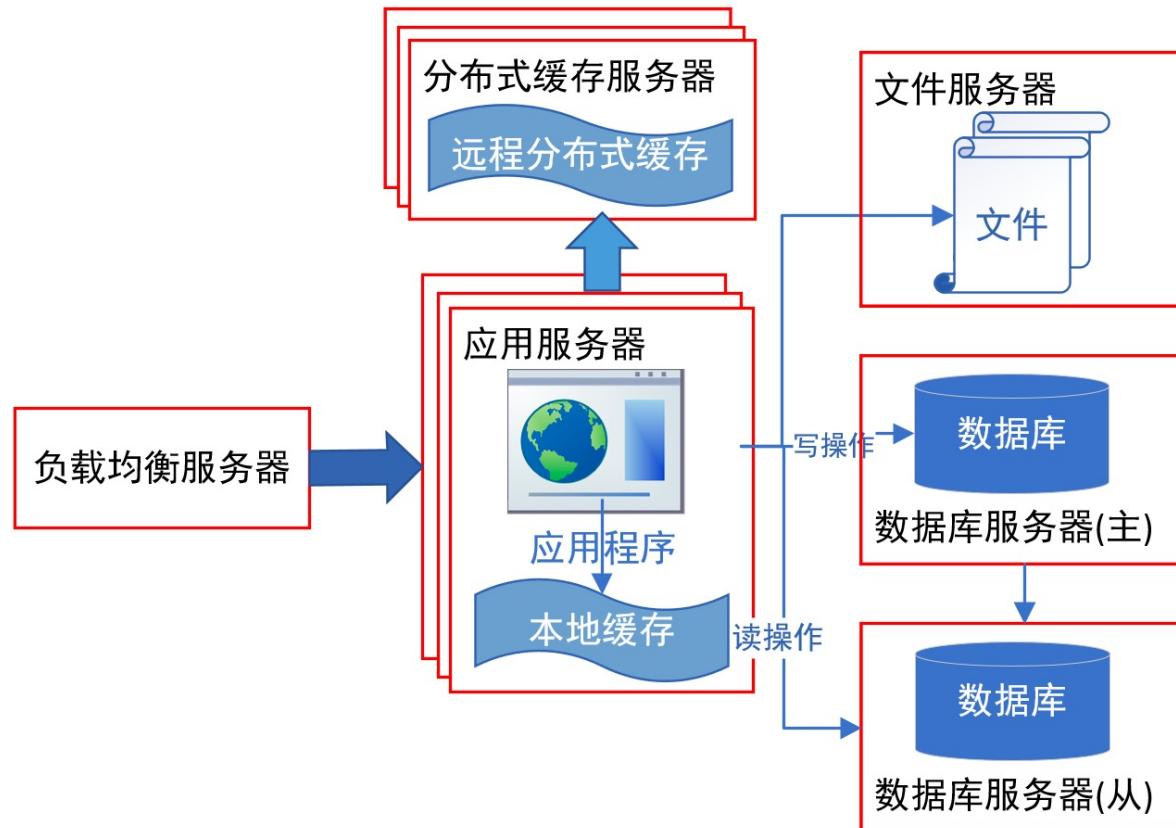
# Architecture of Java Web Application



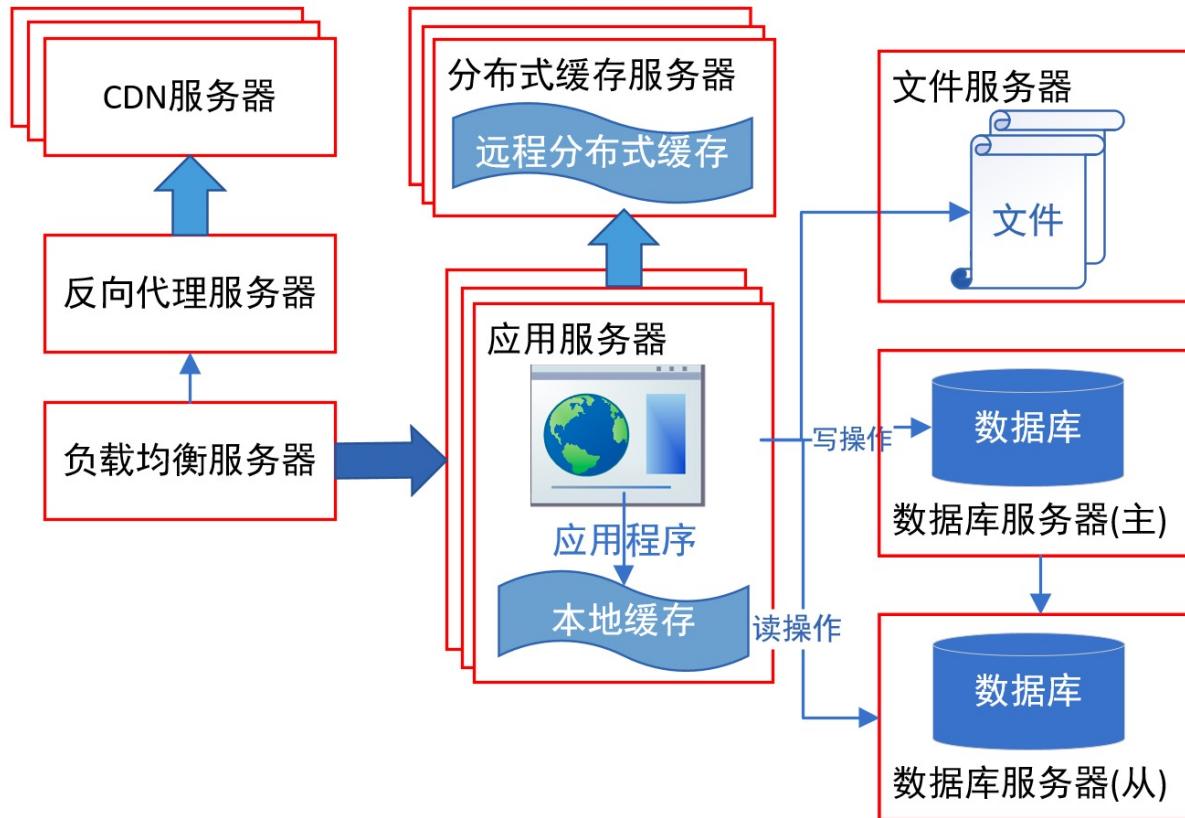
# Architecture of Java Web Application



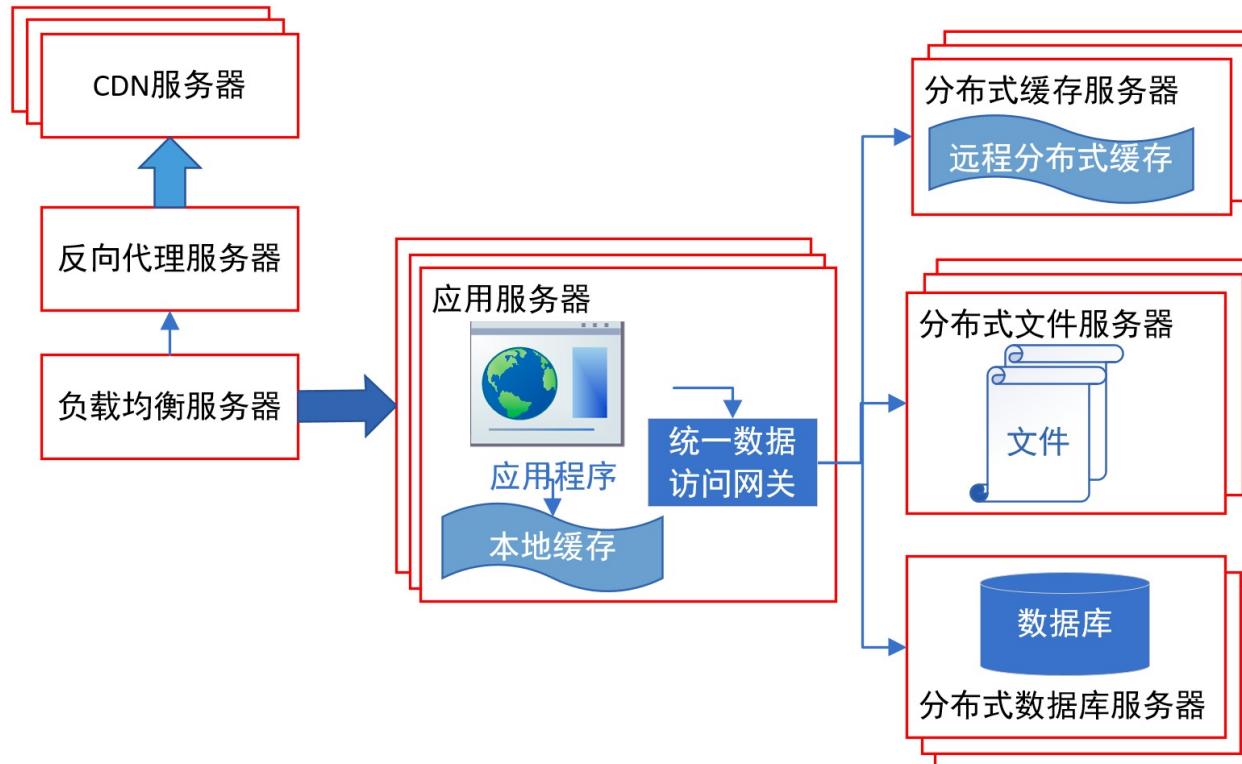
# Architecture of Java Web Application



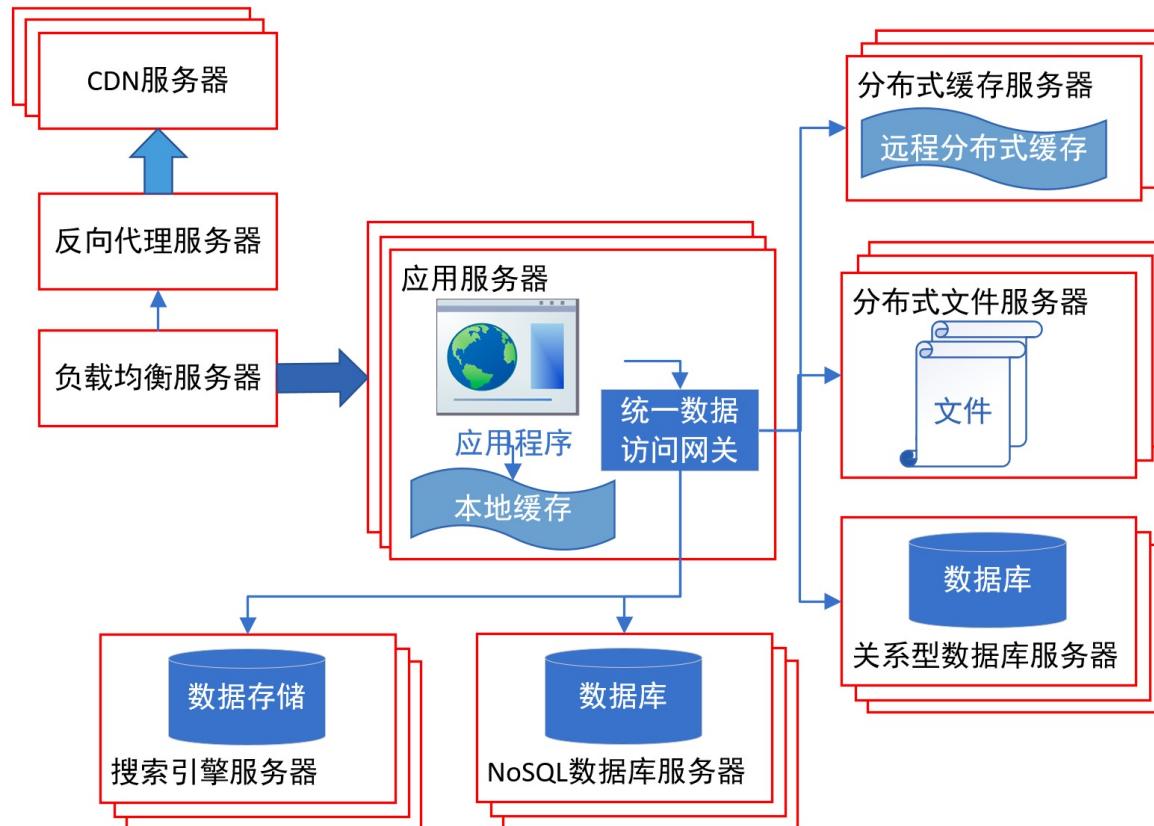
# Architecture of Java Web Application



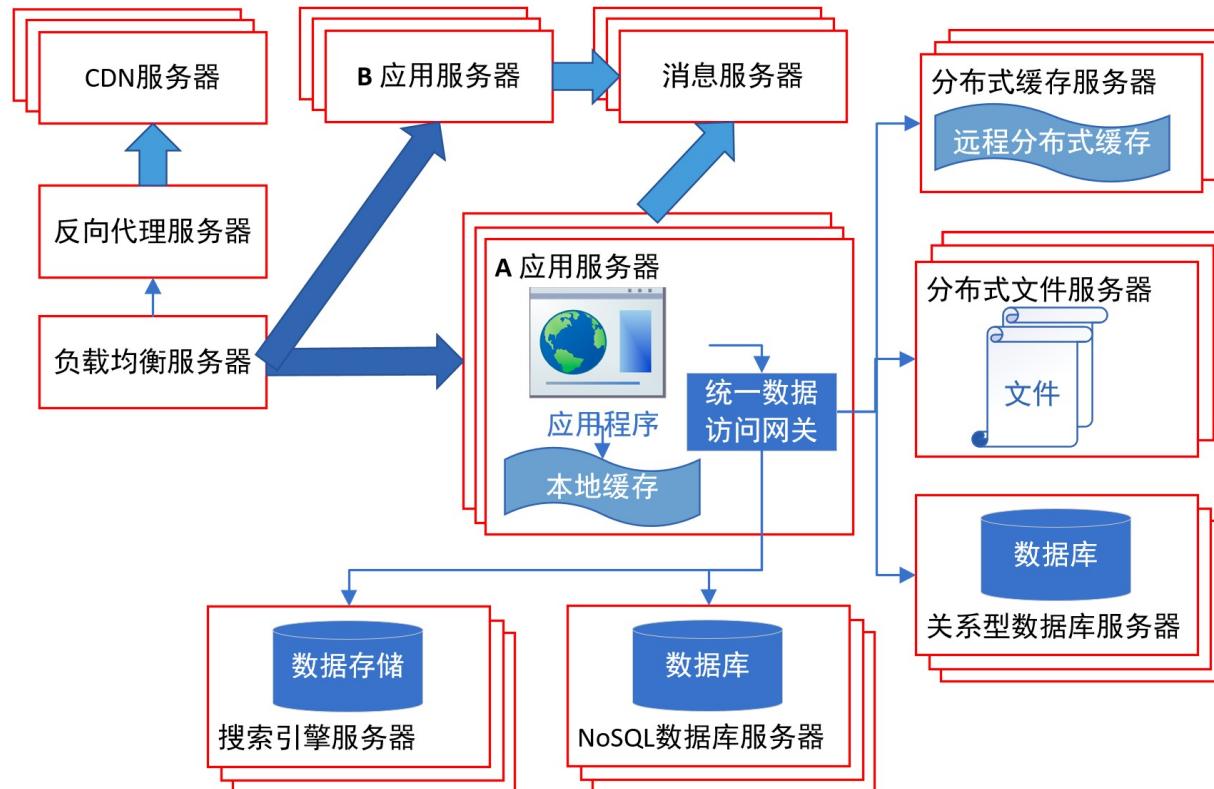
# Architecture of Java Web Application



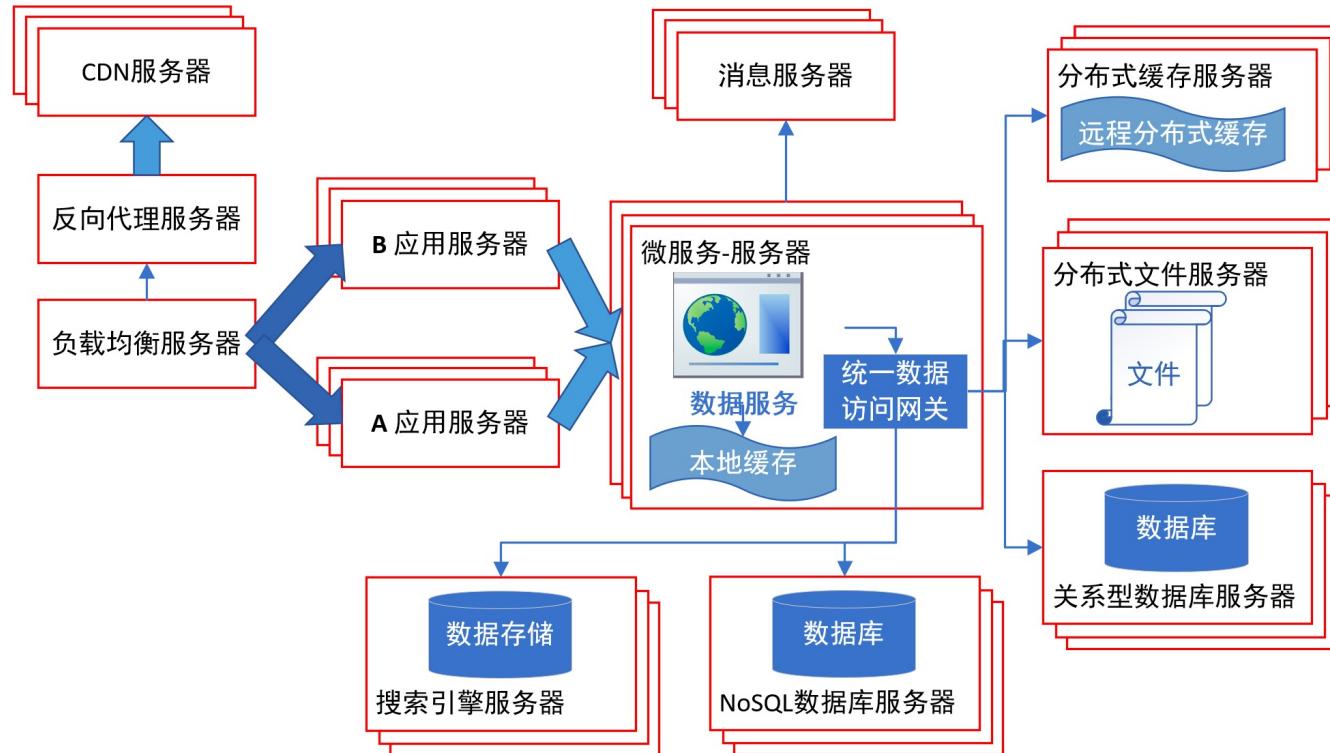
# Architecture of Java Web Application



# Architecture of Java Web Application



# Architecture of Java Web Application



# Architecture of Enterprise Applications 1

## Stateful and Stateless Services

Haopeng Chen

*REliable, INtelligent and Scalable Systems Group (REINS)*

Shanghai Jiao Tong University

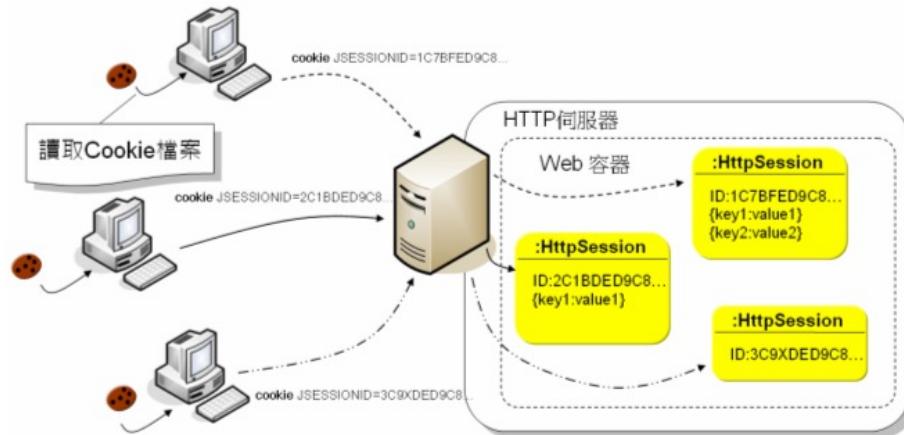
Shanghai, China

<http://reins.se.sjtu.edu.cn/~chenhp>

e-mail: chen-hp@sjtu.edu.cn

# Recall: HTTP session state

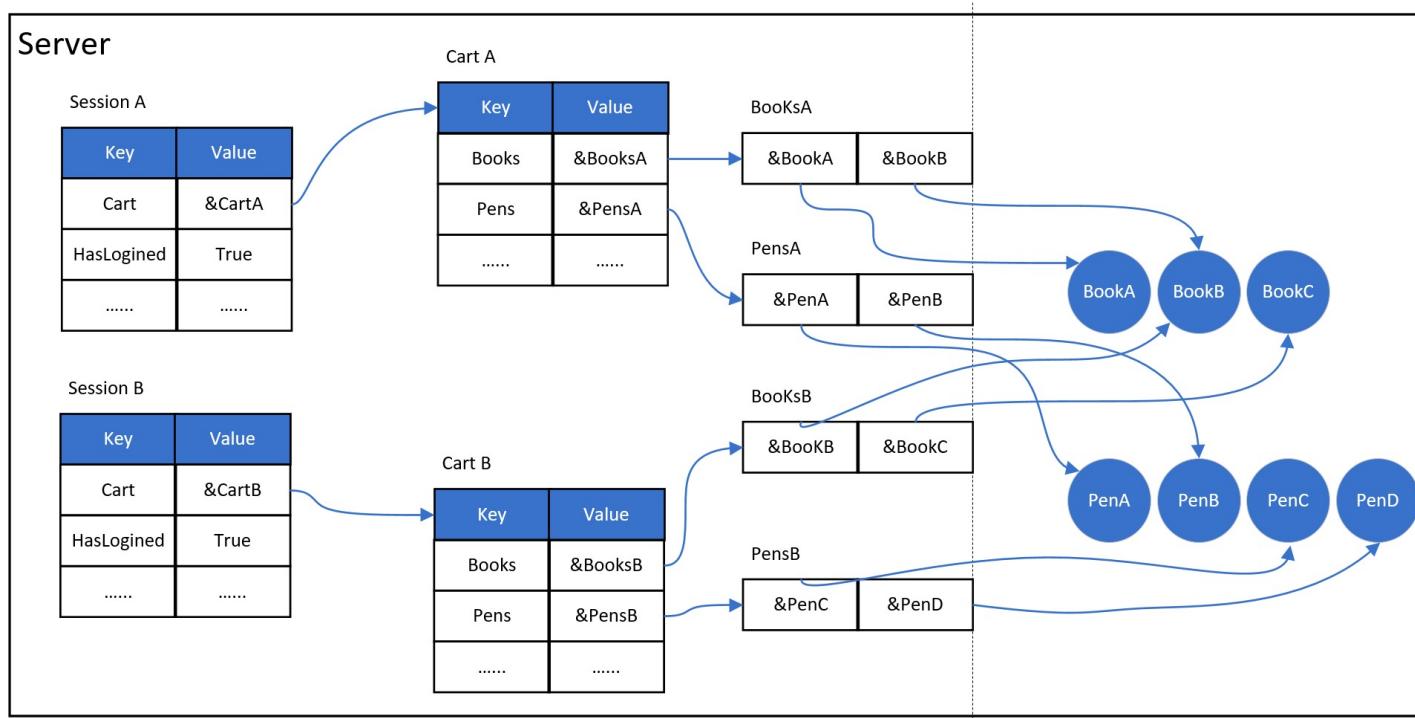
- HTTP is a **stateless** protocol
  - A stateless protocol does not require the HTTP server to retain information or status about each user for the duration of multiple requests.



– from: <https://www.openhome.cc/Gossip/ServletJSP/BehindHttpSession.html>

# Recall: HTTP session state

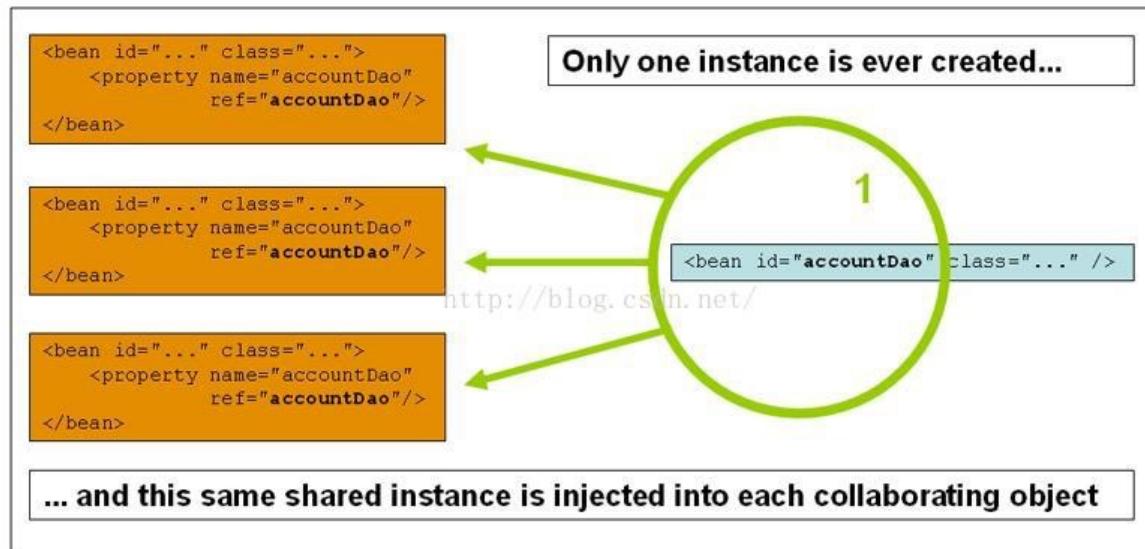
- HTTP is a **stateless** protocol



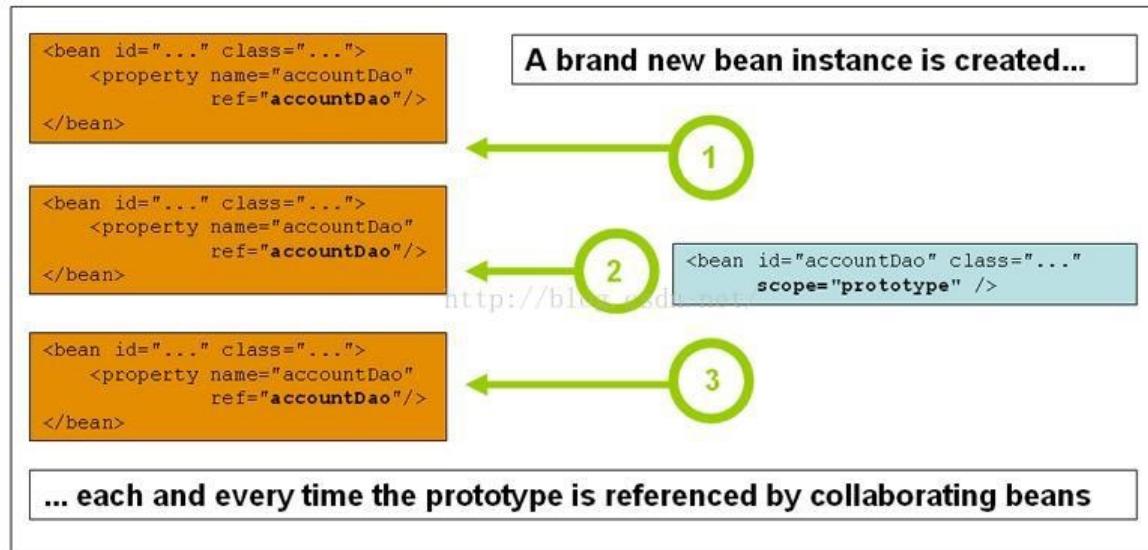
- What's the meaning of scope?

Scope	Description
singleton	(Default) Scopes a single bean definition to a single object instance for each Spring IoC container.
prototype	Scopes a single bean definition to any number of object instances.
request	Scopes a single bean definition to the lifecycle of a single HTTP request. That is, each HTTP request has its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring ApplicationContext .
session	Scopes a single bean definition to the lifecycle of an HTTP Session . Only valid in the context of a web-aware Spring ApplicationContext .
application	Scopes a single bean definition to the lifecycle of a ServletContext . Only valid in the context of a web-aware Spring ApplicationContext .
websocket	Scopes a single bean definition to the lifecycle of a WebSocket . Only valid in the context of a web-aware Spring ApplicationContext .

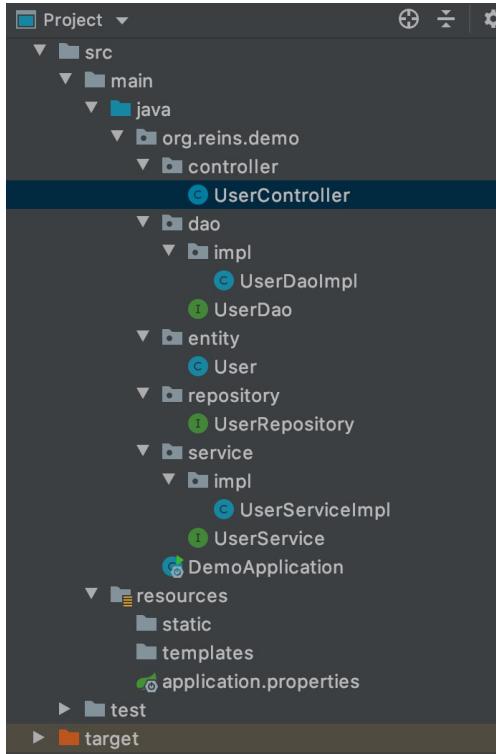
- Singleton : Stateless



- Prototype : Stateful



# Default Scope - singleton



UserController.java

```
@RestController
public class UserController {

    @Autowired
    private UserService userService;

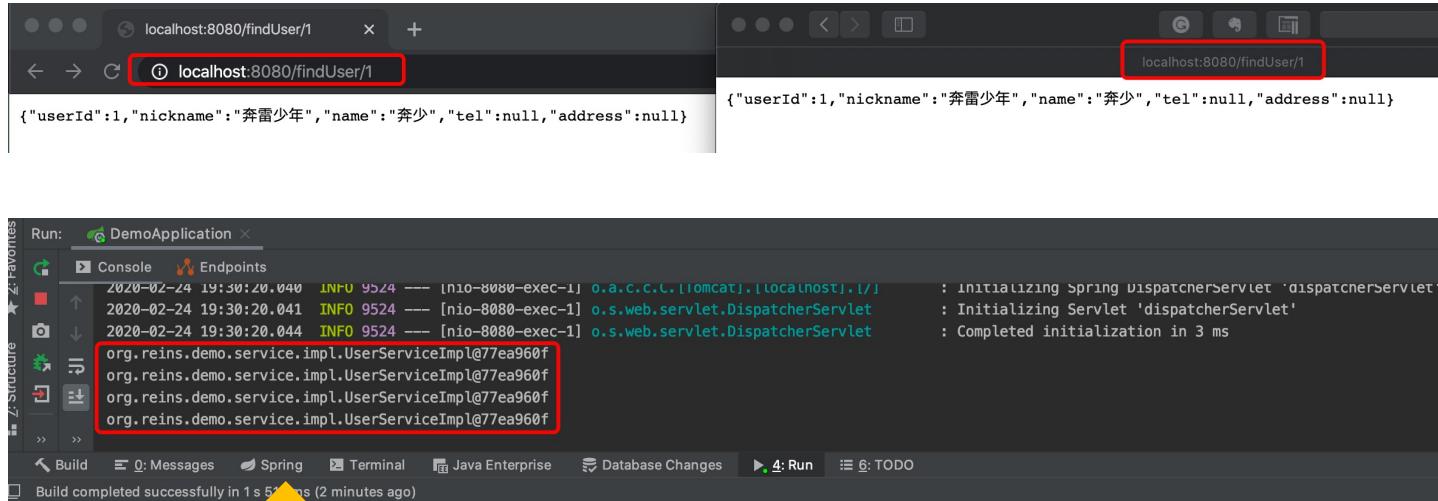
    @GetMapping(value = "/findUser/{id}")
    public User findOne(@PathVariable("id") String id) {
        System.out.println(userService);
        return userService.findUserById(Integer.valueOf(id));
    }
}
```

UserServiceImpl.java

```
@Service
public class UserServiceImpl implements UserService {
    @Autowired
    private UserDao userDao;

    @Override
    public User findUserById(Integer id) {
        return userDao.findOne(id);
    }
}
```

# Default Scope - singleton



Single Service instance

# Scope - prototype

```
UserController.java
@RestController
public class UserController {

    @Autowired
    private UserService userService;

    @GetMapping(value = "/findUser/{id}")
    public User findOne(@PathVariable("id") String id) {
        System.out.println(userService);
        return userService.findUserById(Integer.valueOf(id));
    }
}
```

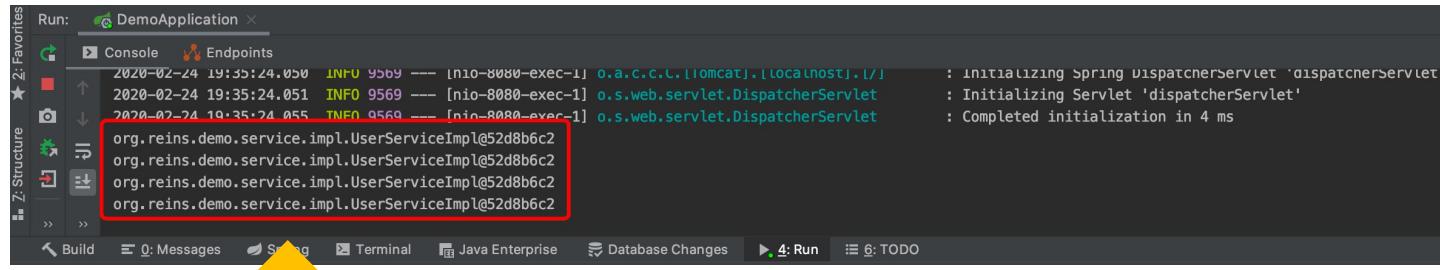
prototype



```
UserServiceImpl.java
@Service
@Scope("prototype")
public class UserServiceImpl implements UserService {
    @Autowired
    private UserDao userDao;

    @Override
    public User findUserById(Integer id) {
        return userDao.findOne(id);
    }
}
```

# Scope - prototype



Still Single. Why?

# Scope - prototype

prototype

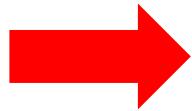


```
UserController.java
@RestController
@Scope("prototype")
public class UserController {

    @Autowired
    private UserService userService;

    @GetMapping(value = "/findUser/{id}")
    public User findOne(@PathVariable("id") String id) {
        System.out.println(userService);
        return userService.findUserById(Integer.valueOf(id));
    }
}
```

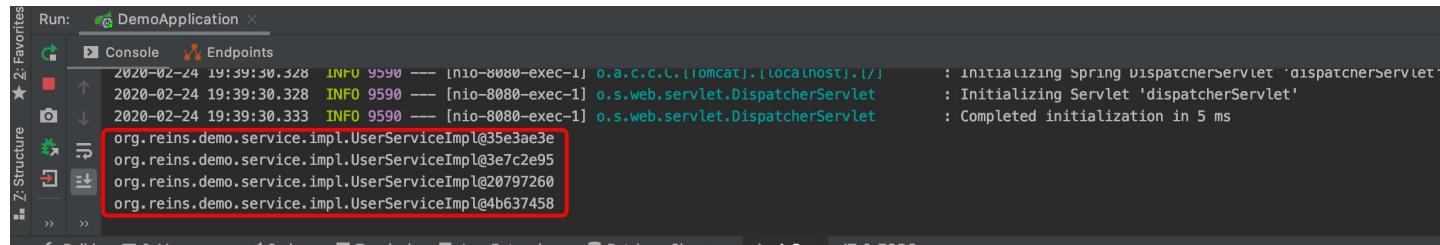
prototype



```
UserServiceImpl.java
@Service
@Scope("prototype")
public class UserServiceImpl implements UserService {
    @Autowired
    private UserDao userDao;

    @Override
    public User findUserById(Integer id) {
        return userDao.findOne(id);
    }
}
```

# Scope - prototype



Now different. But not  
good!

# Scope - prototype

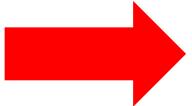
WebApplicationContext



```
UserController.java
@RestController
public class UserController {
    @Autowired
    WebApplicationContext applicationContext;

    @GetMapping(value = "/findUser/{id}")
    public User findOne(@PathVariable("id") String id) {
        UserService userService = applicationContext.getBean(UserService.class);
        System.out.println(userService);
        return userService.findUserById(Integer.valueOf(id));
    }
}
```

UserService



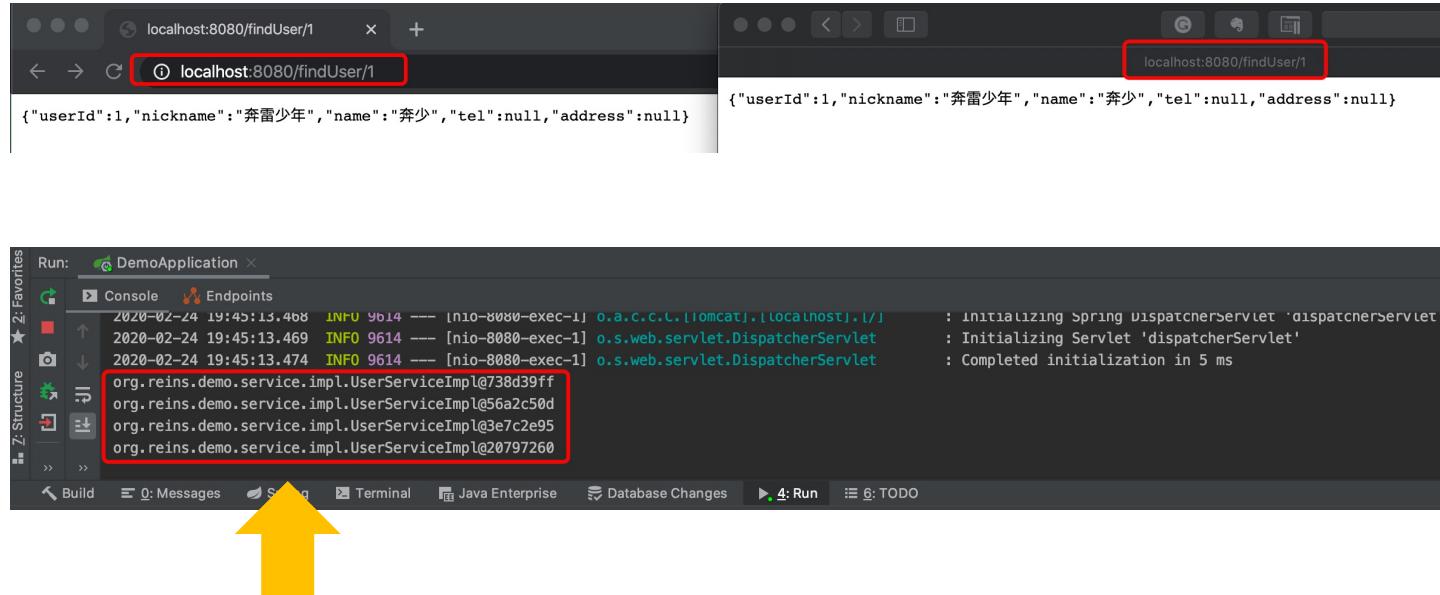
prototype



```
UserServiceImpl.java
@Service
@Scope("prototype")
public class UserServiceImpl implements UserService {
    @Autowired
    private UserDao userDao;

    @Override
    public User findUserById(Integer id) {
        return userDao.findOne(id);
    }
}
```

# Scope - prototype



Now different. Why is this solution better?

# Scope - session

WebApplicationContext



```
UserController.java
@RestController
public class UserController {
    @Autowired
    WebApplicationContext applicationContext;

    @GetMapping(value = "/findUser/{id}")
    public User findOne(@PathVariable("id") String id) {
        UserService userService = applicationContext.getBean(UserService.class);
        System.out.println(userService);
        return userService.findUserById(Integer.valueOf(id));
    }
}
```

UserService



session



```
UserServiceImpl.java
@Service
@Scope("session")
public class UserServiceImpl implements UserService {
    @Autowired
    private UserDao userDao;

    @Override
    public User findUserById(Integer id) {
        return userDao.findOne(id);
    }
}
```

# Scope - session



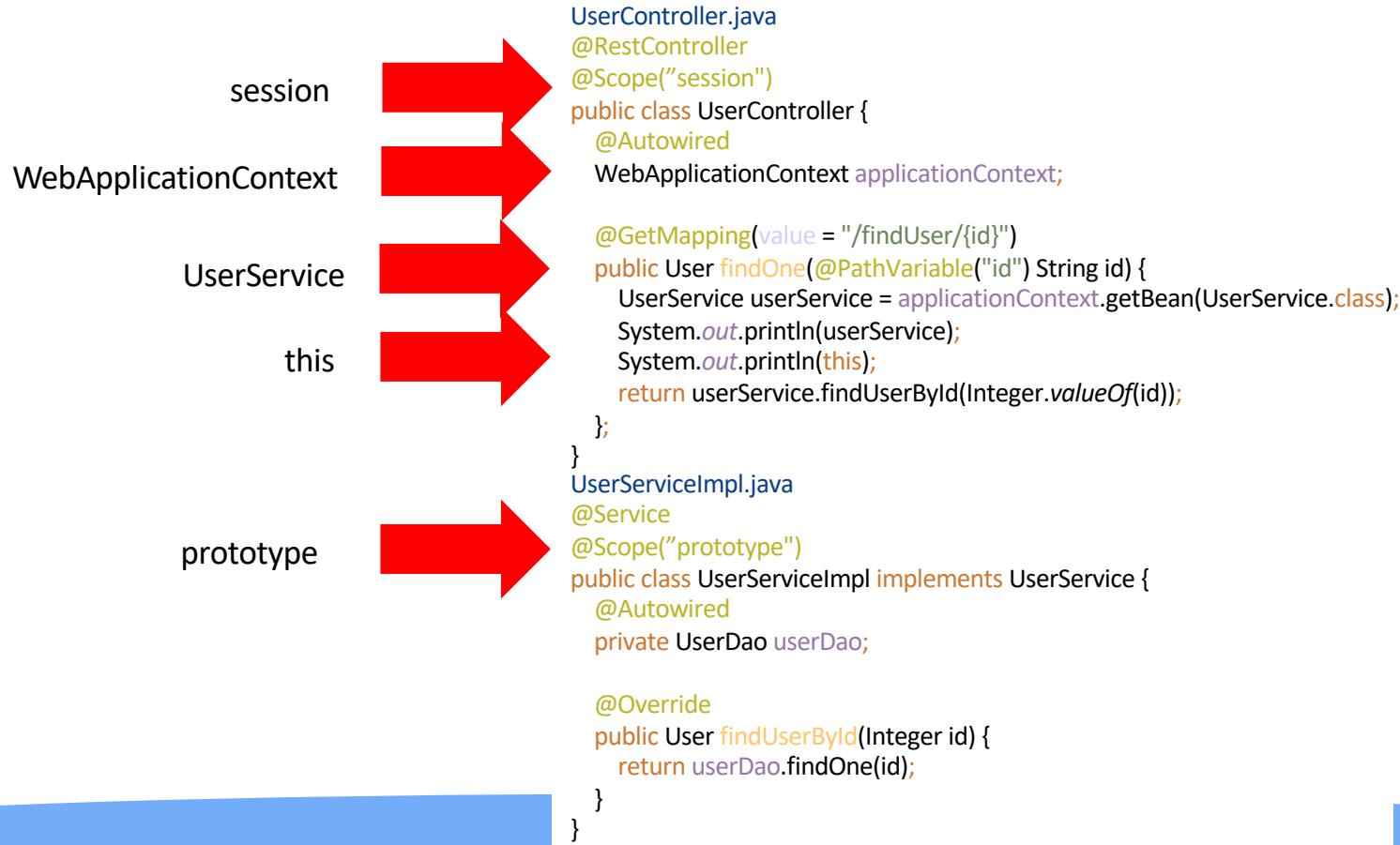
A screenshot of a Java IDE's run console. The title bar says "Run: DemoApplication". The left sidebar shows "Favorites" and "Structure". The main area shows log output:

```
2020-02-24 19:47:26.689 INFO 9623 --- [nio-8080-exec-1] o.a.c.c.l.tomcat.[localhost].1/ : initializing Spring DispatcherServlet 'dispatcherServlet'  
2020-02-24 19:47:26.690 INFO 9623 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'  
2020-02-24 19:47:26.694 INFO 9623 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 4 ms  
org.reins.demo.service.impl.UserServiceImpl@4626fb9c  
org.reins.demo.service.impl.UserServiceImpl@4626fb9c  
org.reins.demo.service.impl.UserServiceImpl@4b637458  
org.reins.demo.service.impl.UserServiceImpl@4b637458
```

A large yellow arrow points upwards from the list of service instances in the log to the "Structure" tab in the sidebar.

Two Service instances

# Scope - session



# Scope - session

The screenshot shows a Java IDE interface with two browser tabs and a log viewer.

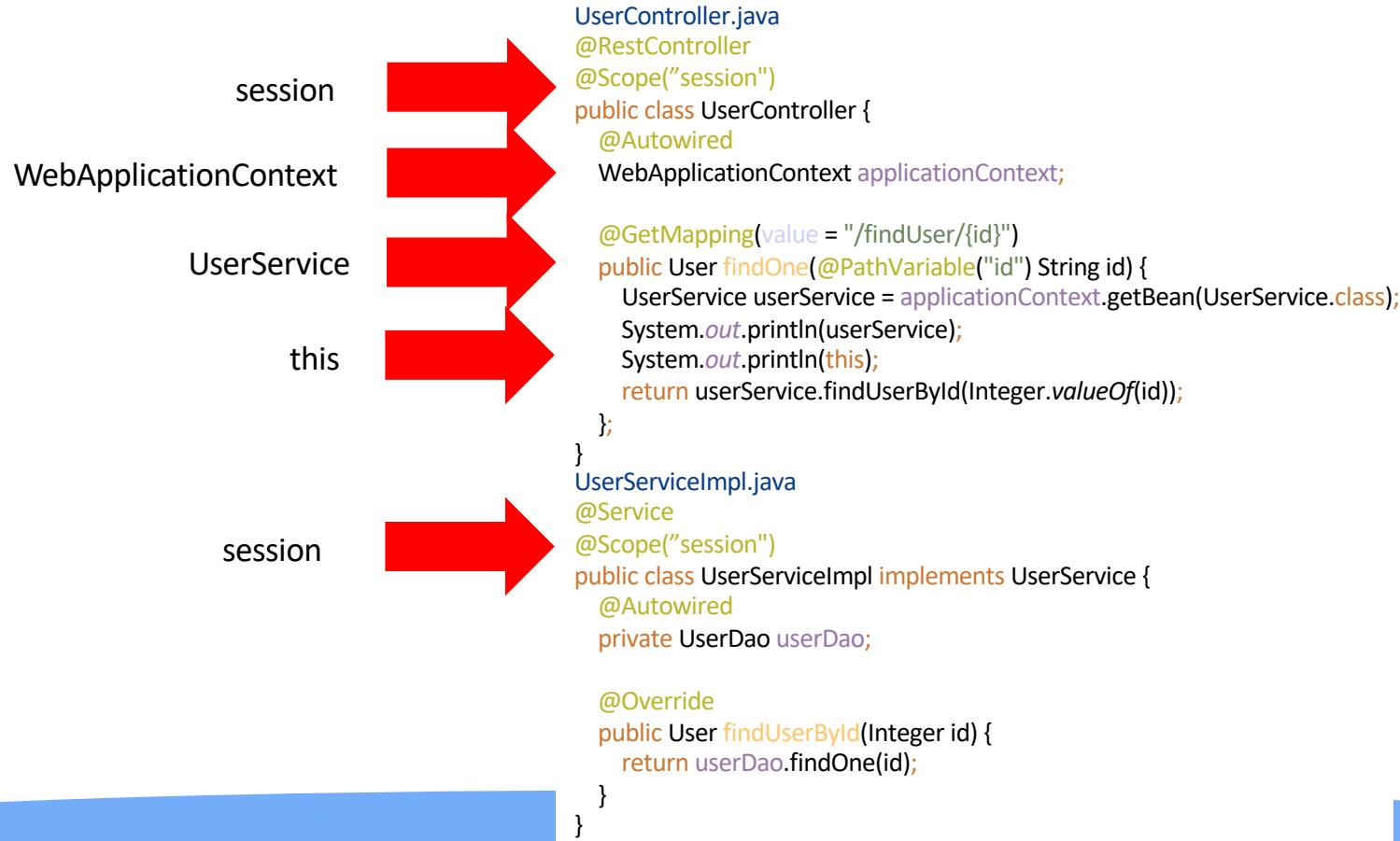
- Browser Tabs:** Both tabs show the URL `localhost:8080/findUser/1`. The left tab displays the JSON response: `{"userId":1,"nickname":"奔雷少年","name":"奔少","tel":null,"address":null}`. The right tab also displays the same JSON response.
- Log Viewer:** The log viewer shows two INFO-level log entries from the `DispatcherServlet`:

```
2020-02-24 19:52:30.980 INFO 9686 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'  
2020-02-24 19:52:30.984 INFO 9686 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 4 ms
```
- Service Instances:** A red box highlights the stack traces for the service instances. The left trace is for `UserServiceImpl@2c0b1967` and the right trace is for `UserServiceImpl@20797260`.



Two Controller instances  
Four Service instances

# Scope - session



# Scope - session

The screenshot shows a Java IDE interface with two browser tabs and a terminal window.

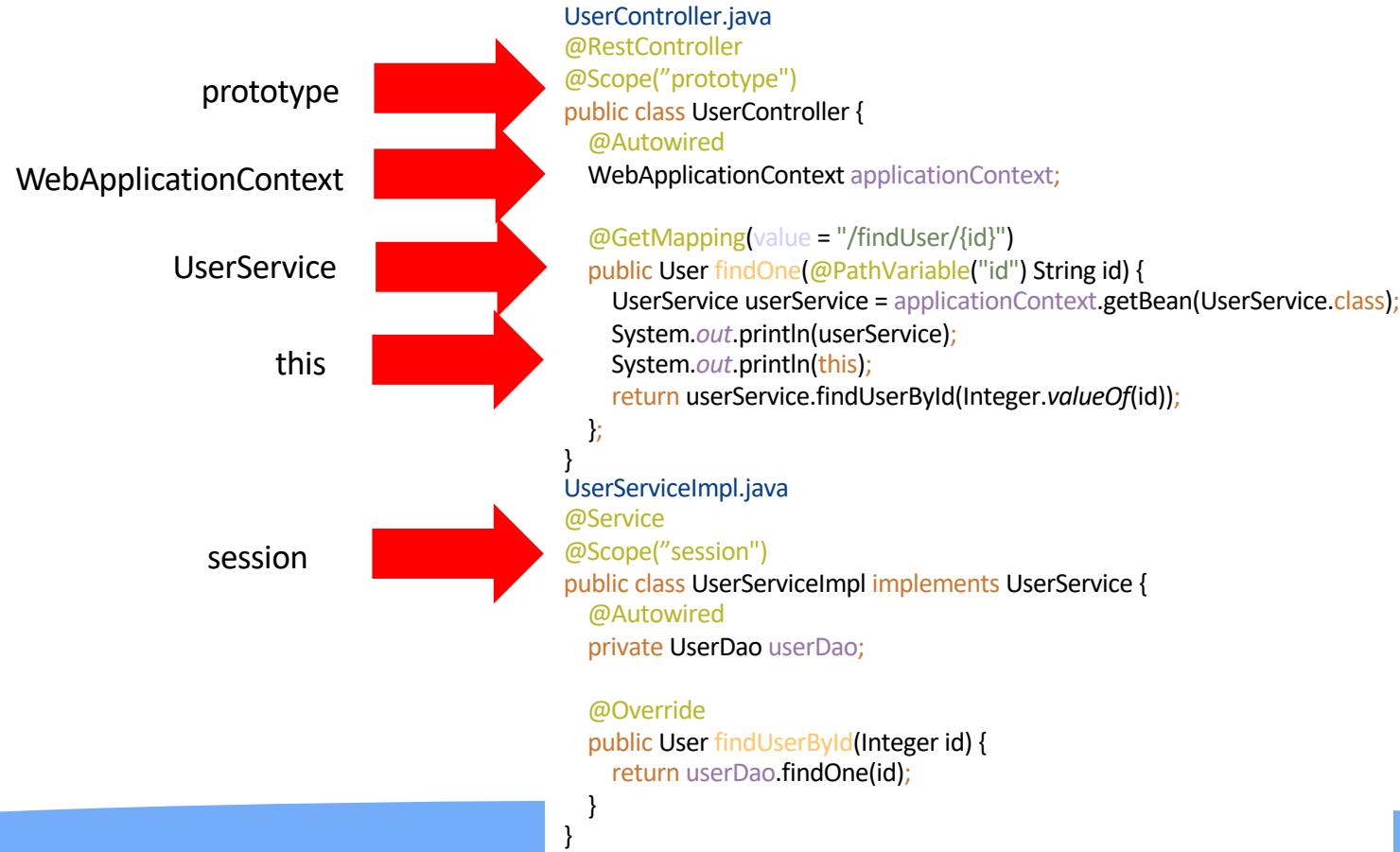
- Browser Tabs:** Both tabs are titled "localhost:8080/findUser/1". The left tab displays the JSON response: {"userId":1,"nickname":"奔雷少年","name":"奔少","tel":null,"address":null}. The right tab also displays the same JSON response.
- Terminal Window:** The terminal shows the application logs:

```
2020-02-24 19:56:49.974 INFO 9852 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2020-02-24 19:56:49.979 INFO 9852 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 5 ms
org.reins.demo.service.impl.UserServiceImpl@55692aa0
org.reins.demo.controller.UserController@5a7d17dd
org.reins.demo.service.impl.UserServiceImpl@55692aa0
org.reins.demo.controller.UserController@5a7d17dd
org.reins.demo.service.impl.UserServiceImpl@2a811910
org.reins.demo.controller.UserController@2ebe981f
org.reins.demo.service.impl.UserServiceImpl@2a811910
org.reins.demo.controller.UserController@2ebe981f
```

A yellow arrow points from the text "Two Controller instances" to the first two lines of the log.
- Toolbars and Status:** The bottom toolbar includes icons for Build, Messages, Spring, Terminal, Java Enterprise, Database Changes, Run, and TODO. The status bar shows the number 32.

Two Controller instances  
Two Service instances

# Scope - session



# Scope - session

The screenshot shows a Java development environment with two browser tabs and a terminal window.

- Browser Tab 1:** localhost:8080/findUser/1. Response: {"userId":1,"nickname":"奔雷少年","name":"奔少","tel":null,"address":null}
- Browser Tab 2:** localhost:8080/findUser/1. Response: {"userId":1,"nickname":"奔雷少年","name":"奔少","tel":null,"address":null}
- Terminal:** Shows log output from 2020-02-24 20:02:16.165. It lists nine controller instances and two service instances, all highlighted with a red box. A yellow arrow points from the text "Four Controller instances" to this red box.

```
2020-02-24 20:02:16.165 INFO 9901 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
: Completed initialization in 4 ms
org.reins.demo.service.impl.UserServiceImpl@4f6300dc
org.reins.demo.controller.UserController@525f1182
org.reins.demo.service.impl.UserServiceImpl@4f6300dc
org.reins.demo.controller.UserController@56b7a24b
org.reins.demo.service.impl.UserServiceImpl@3fb913df
org.reins.demo.controller.UserController@3b8690f0
org.reins.demo.service.impl.UserServiceImpl@3fb913df
org.reins.demo.controller.UserController@586531a9
```

Four Controller instances  
Two Service instances

- 2 Browsers(2 Sessions), 4 Requests( 2 times / session)

		Controller	
		Prototype	session
Service	prototype	4 service instances 4 controller instances	4 service instances 2 controller instances
	session	2 service instances 4 controller instances	2 service instances 2 controller instances

- How should we design the scopes of Beans?

- 请你在大二开发的E-Book系统的基础上，完成下列任务：
  1. 优化服务层设计，确定有状态与无状态服务的运用方式，要求至少分别选择一个服务设计成有状态的和无状态的，并观察它们的差异，确认你的设计合理。
  - 你应该将上述功能集成到你的E-Book系统中，如果你无法将上述功能集成到你的E-Book系统中，可以单独建立工程实现，但是会适当扣分。
  - 请将你编写的相关代码整体压缩后上传，请勿压缩整个工程提交。
- 评分标准：
  1. 选择的有状态服务和无状态服务合理，代码编写正确（2分）

- Web Applications: What are They? What of Them?
  - <http://www.acunetix.com/websitemanagement/web-applications/>
- Web开发的发展史
  - <https://linux.cn/article-3166-1.html>
- Web 研发模式演变
  - <https://github.com/lifesinger/blog/issues/184>
- Java 应用一般架构
  - <https://blog.coding.net/blog/General-architecture-for-Java-applications>
- The IoC Container- Bean Scopes
  - <https://docs.spring.io/spring/docs/5.2.3.RELEASE/spring-framework-reference/core.html#beans-factory-scopes>
- Quick Guide to Spring Bean Scopes
  - <https://www.baeldung.com/spring-bean-scopes>
- Spring注解中@Scope 的使用解说
  - <https://blog.csdn.net/cuichunchi/article/details/79170240>



# Thank You!