Architecture of Enterprise Applications 1
Overview of Enterprise Applications

Haopeng Chen

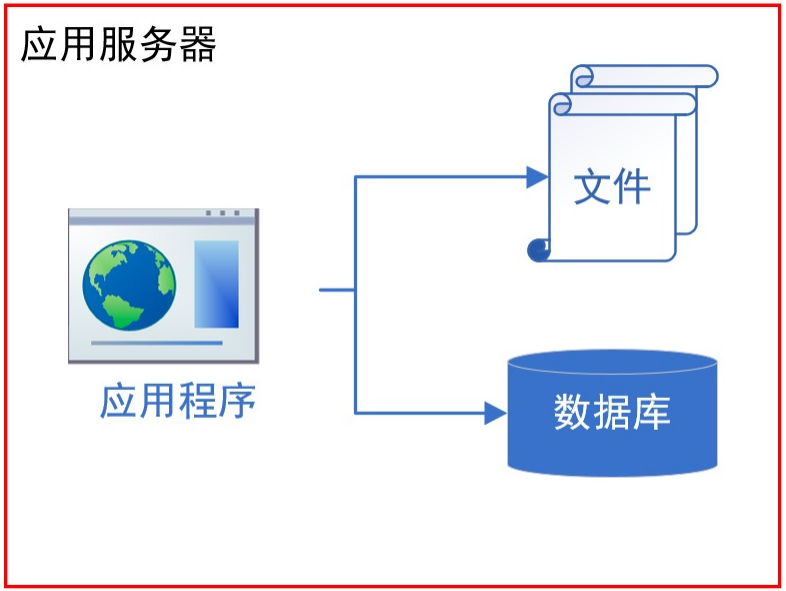
***RE**liable, **IN**telligent and **S**calable Systems Group (**REINS**)*
Shanghai Jiao Tong University
Shanghai, China
http://reins.se.sjtu.edu.cn/~chenhp
e-mail: chen-hp@sjtu.edu.cn

Contents and Objectives

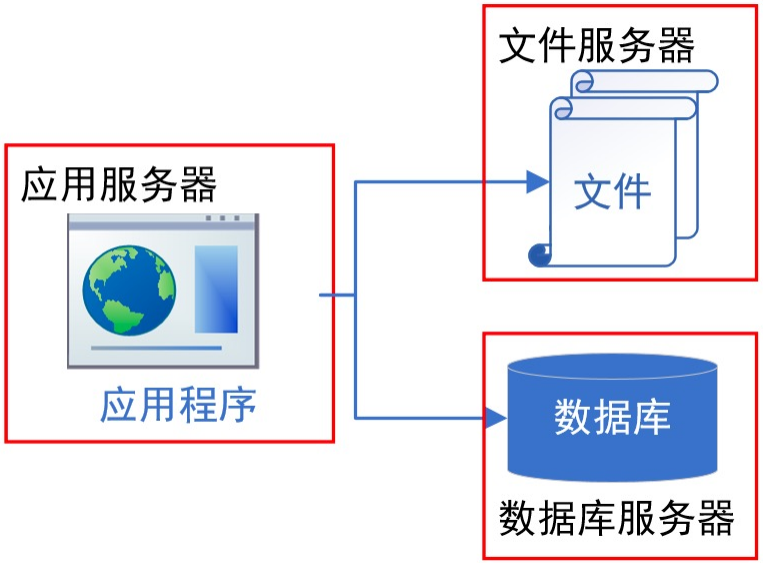
- Contents
  - Architecture
  - Scope

- 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

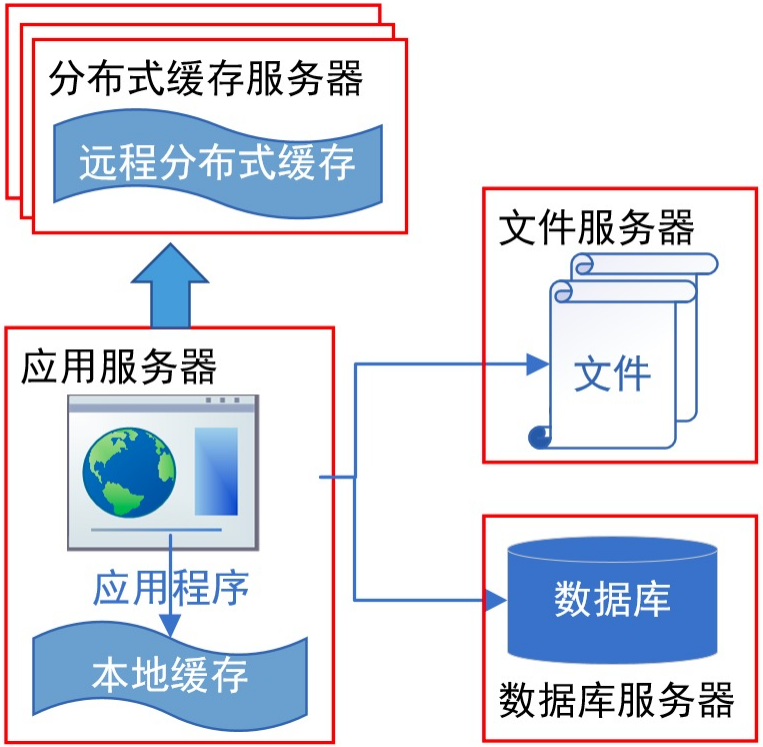
Architecture of Java Web Application



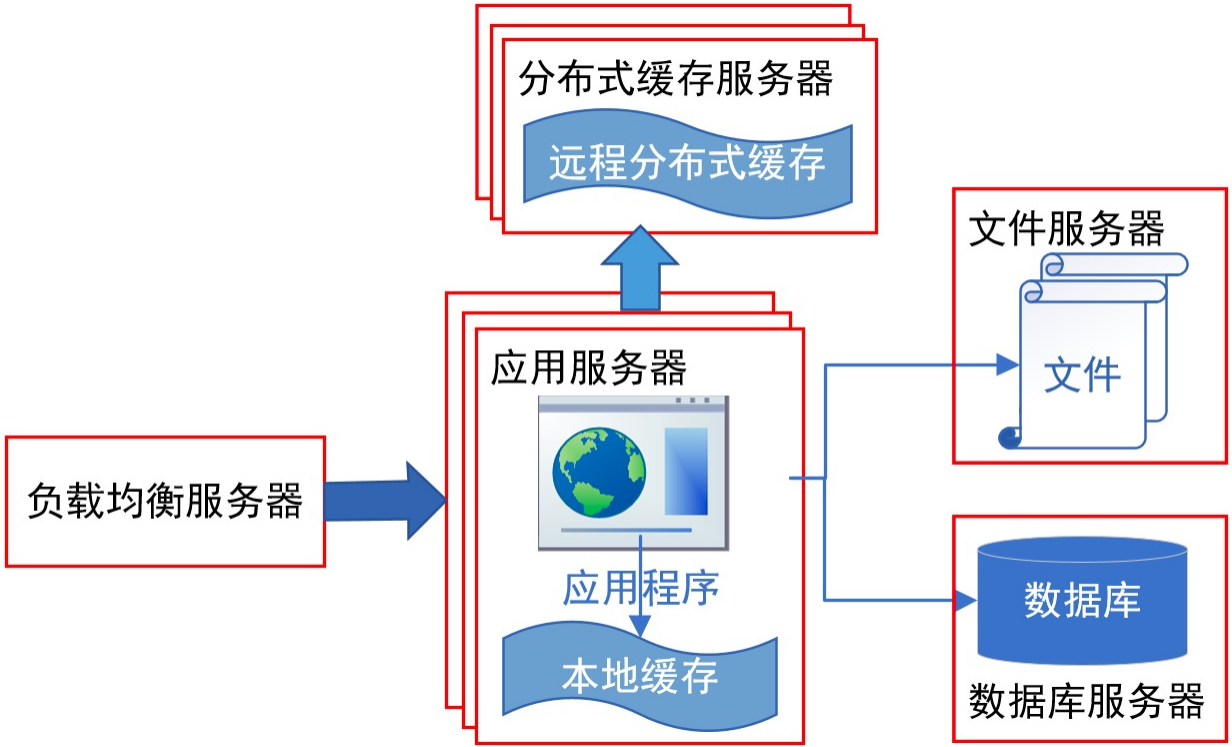
Architecture of Java Web Application



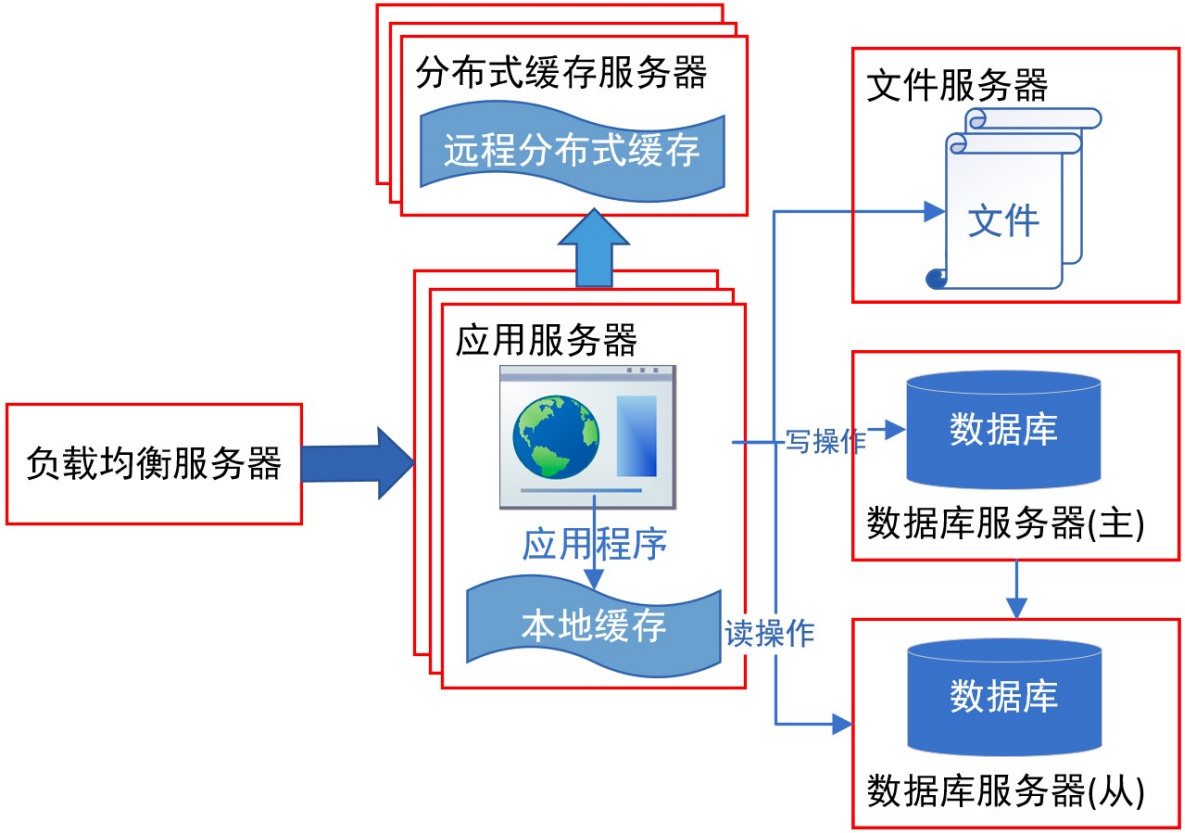
Architecture of Java Web Application



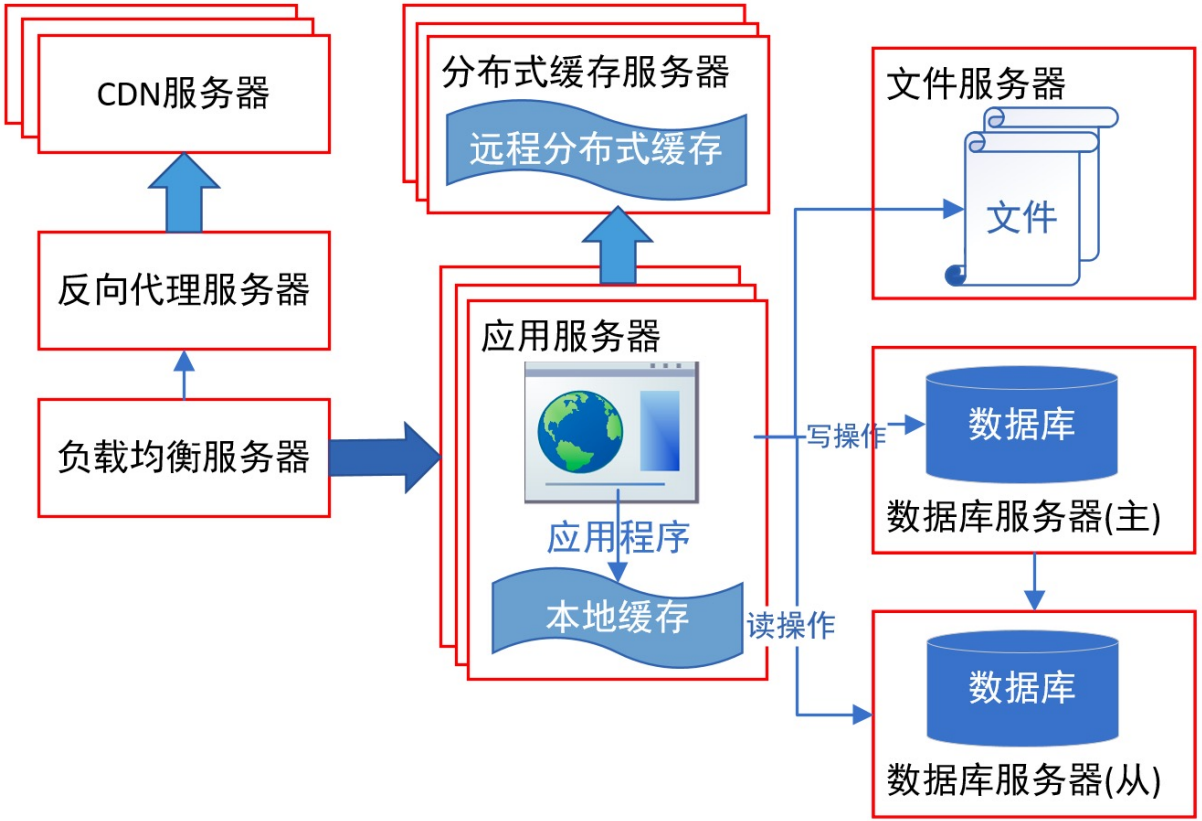
Architecture of Java Web Application



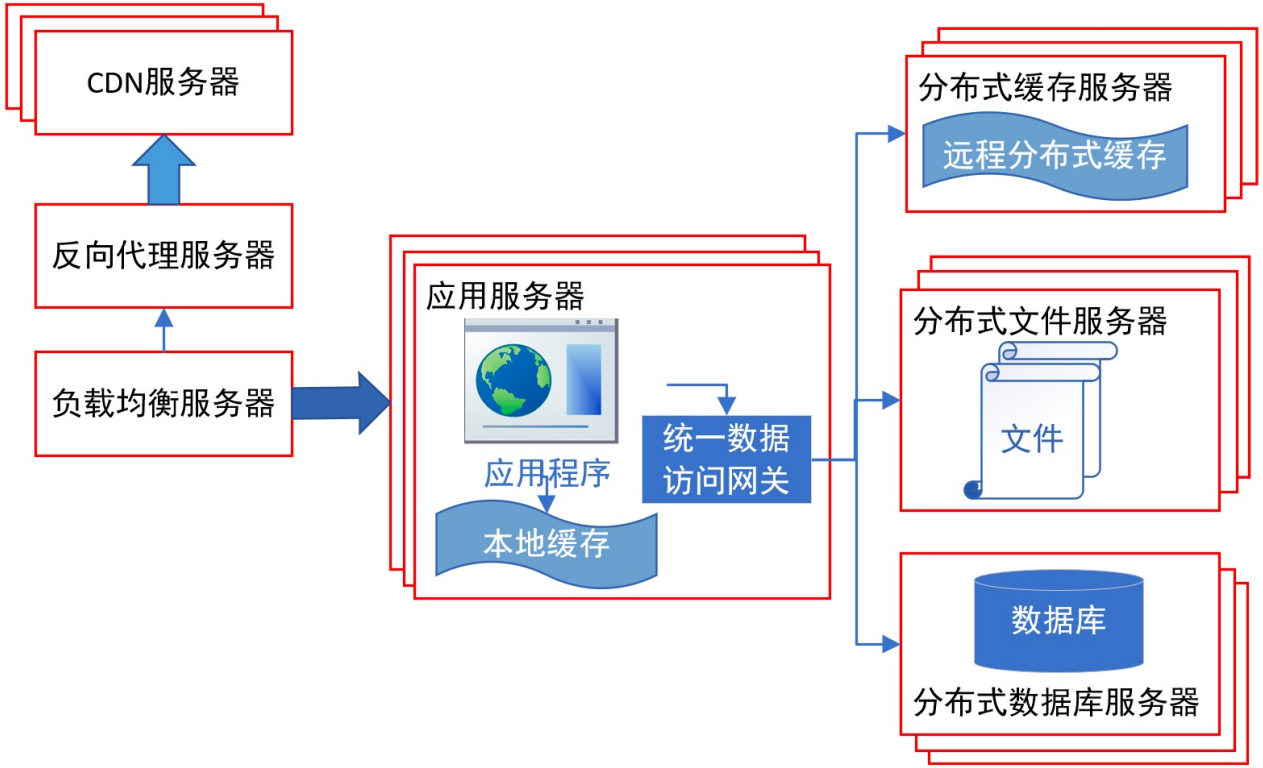
Architecture of Java Web Application



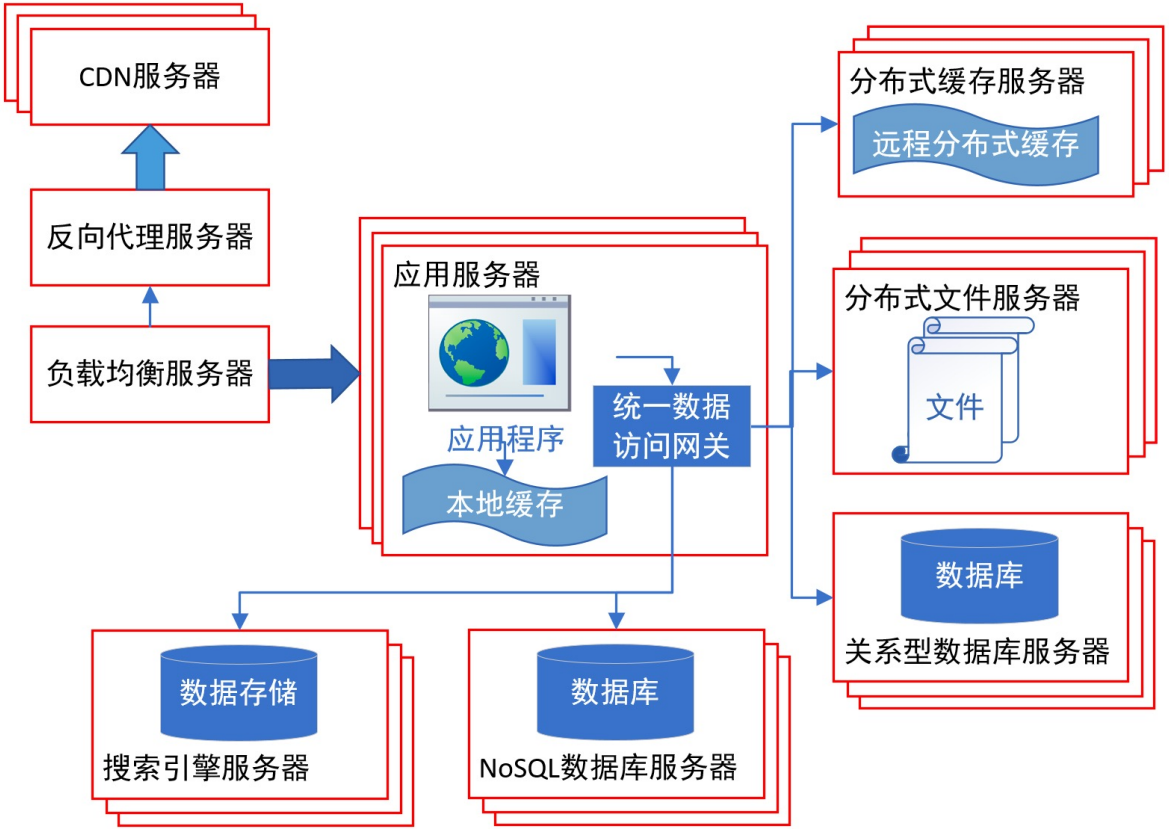
Architecture of Java Web Application



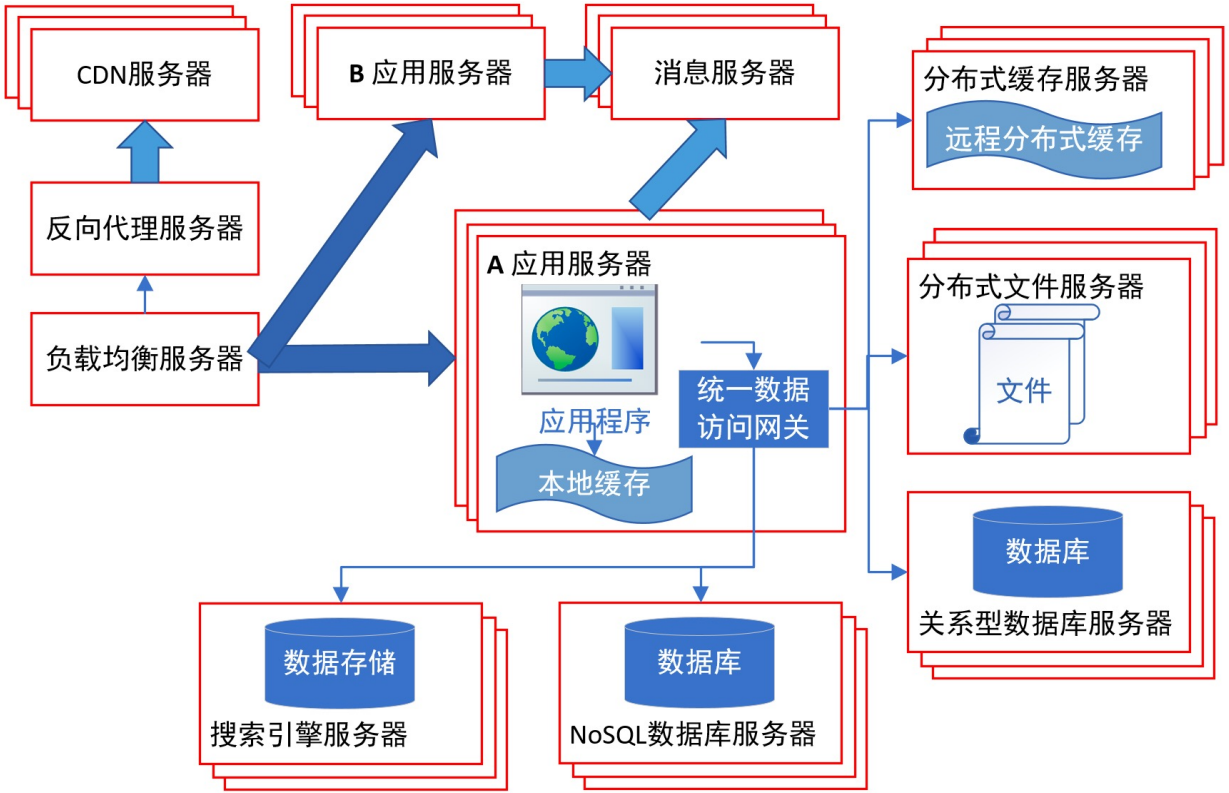
Architecture of Java Web Application



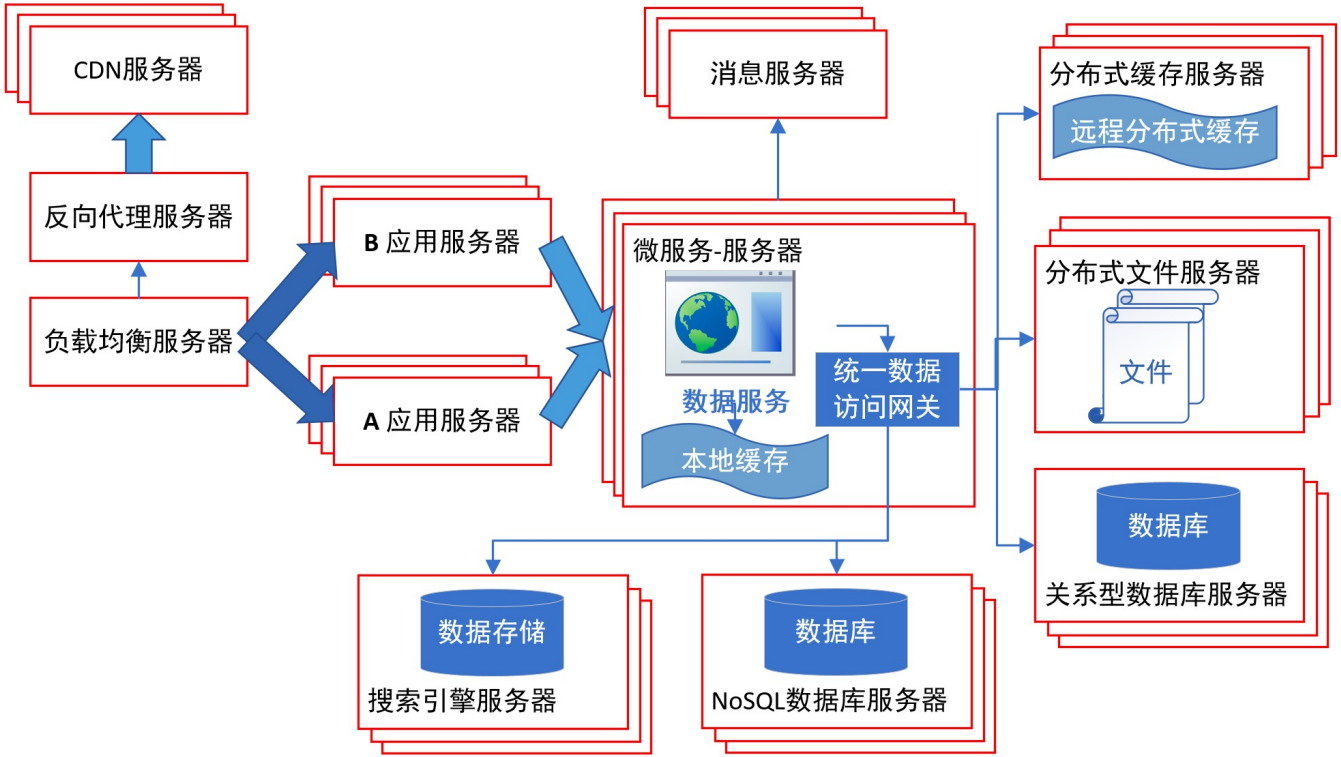
Architecture of Java Web Application



Architecture of Java Web Application



Architecture of Java Web Application



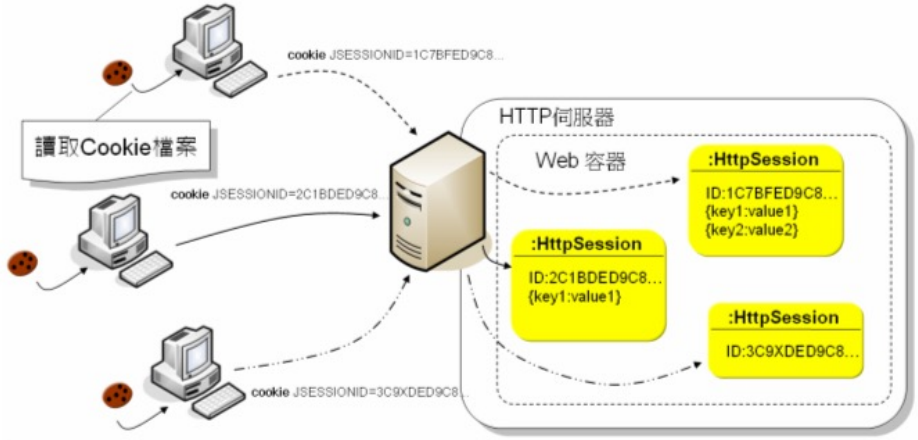
Architecture of Enterprise Applications 1
Stateful and Stateless Services

Haopeng Chen

***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

Recall: HTTP session state

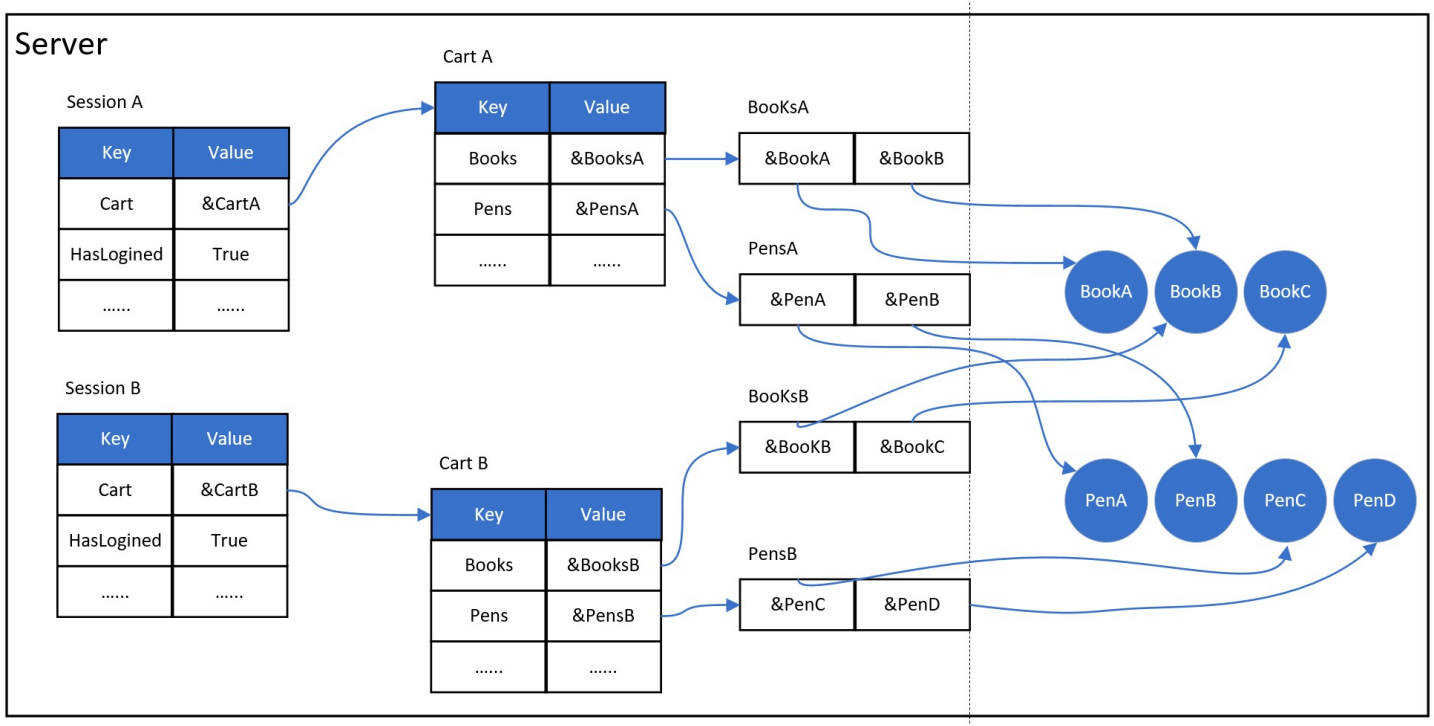
- HTTP is a stateless protocol
  - A stateless protocol does not require the HTTP server to retain information or status about each user for the duration of multiple requests.



  - *from: https://www.openhome.cc/Gossip/ServletJSP/BehindHttpSession.html*

Recall: HTTP session state

- HTTP is a stateless protocol



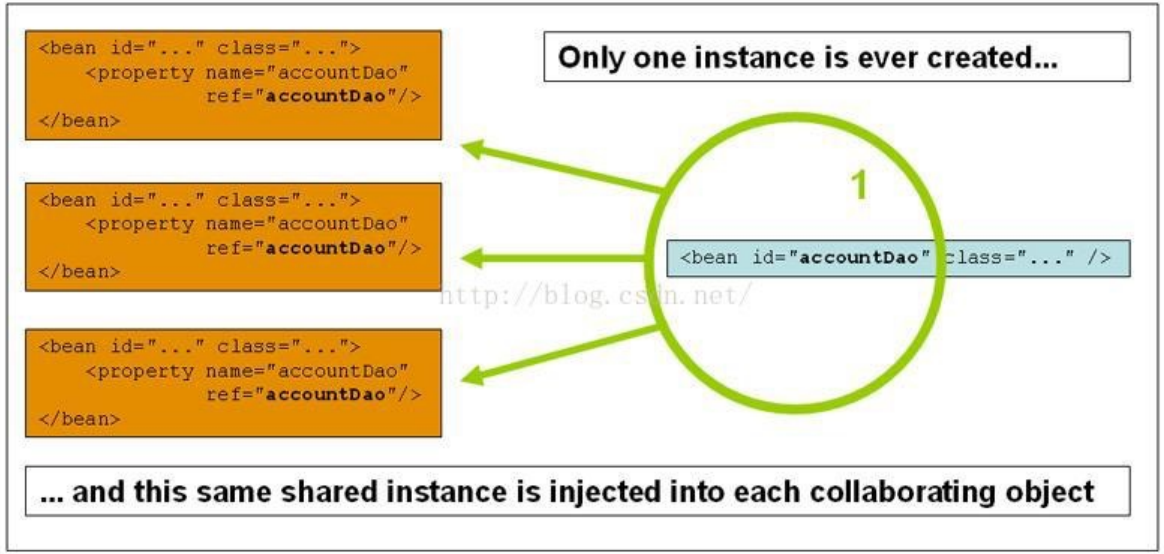
Spring Bean Scopes

- 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`.

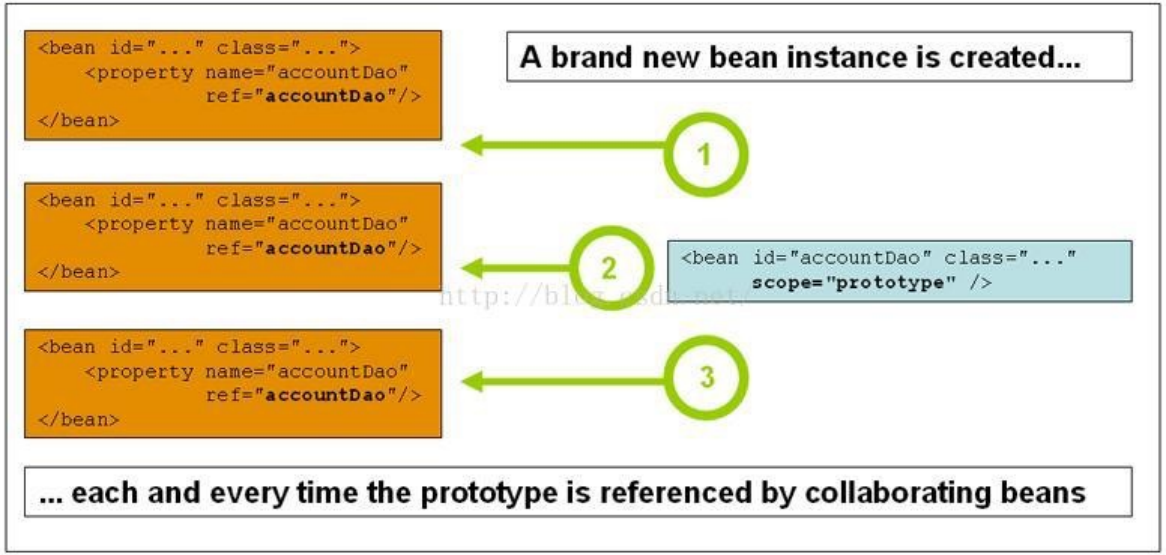
Scope

- Singleton : Stateless

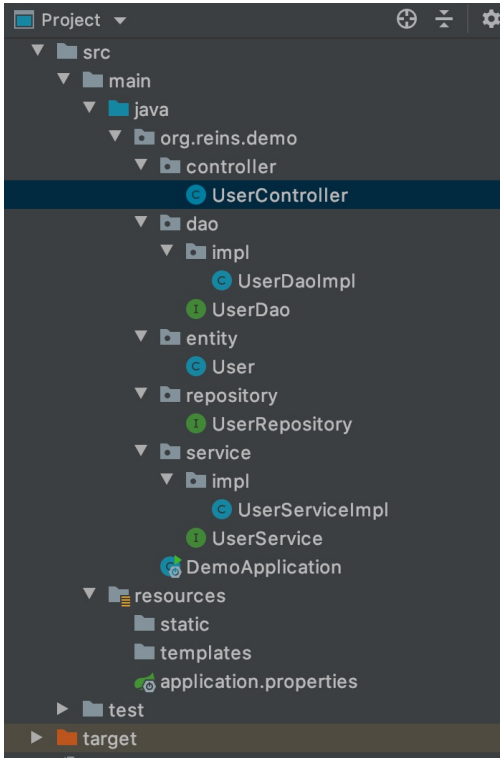


Scope

- Prototype : Stateful



Default Scope - singleton



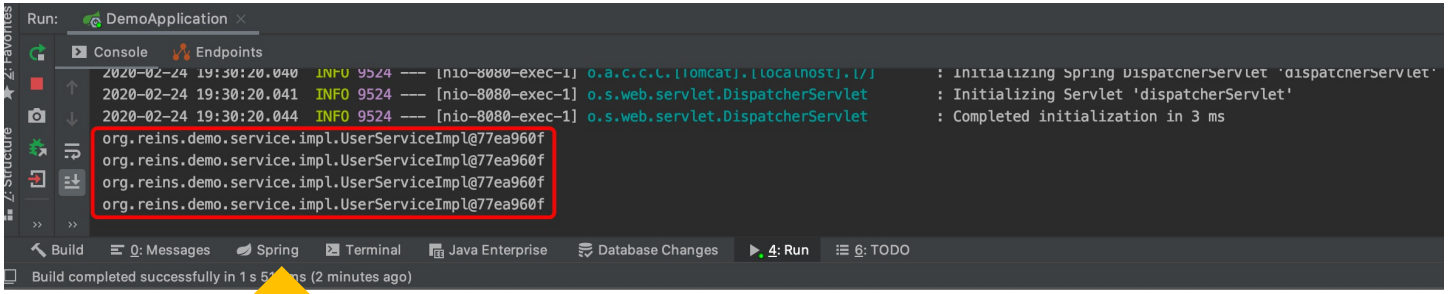
UserController.java

```
@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);
   }
}
```

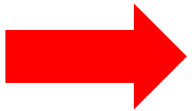
Default Scope - singleton



Single Service instance

Scope - prototype

prototype



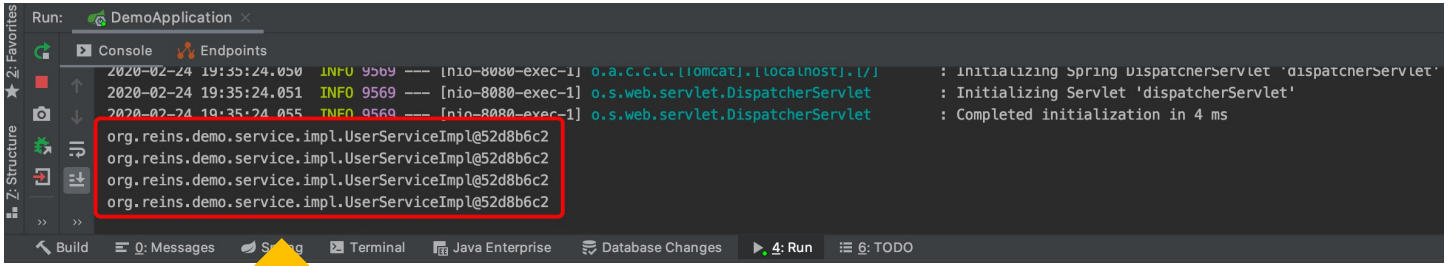
UserController.java

```
@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));
   };
}
```

UserServiceImpl.java

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

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

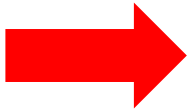
Scope - prototype



Still Single. Why?

Scope - prototype

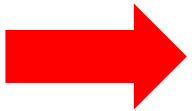
prototype



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));
   };
}
```

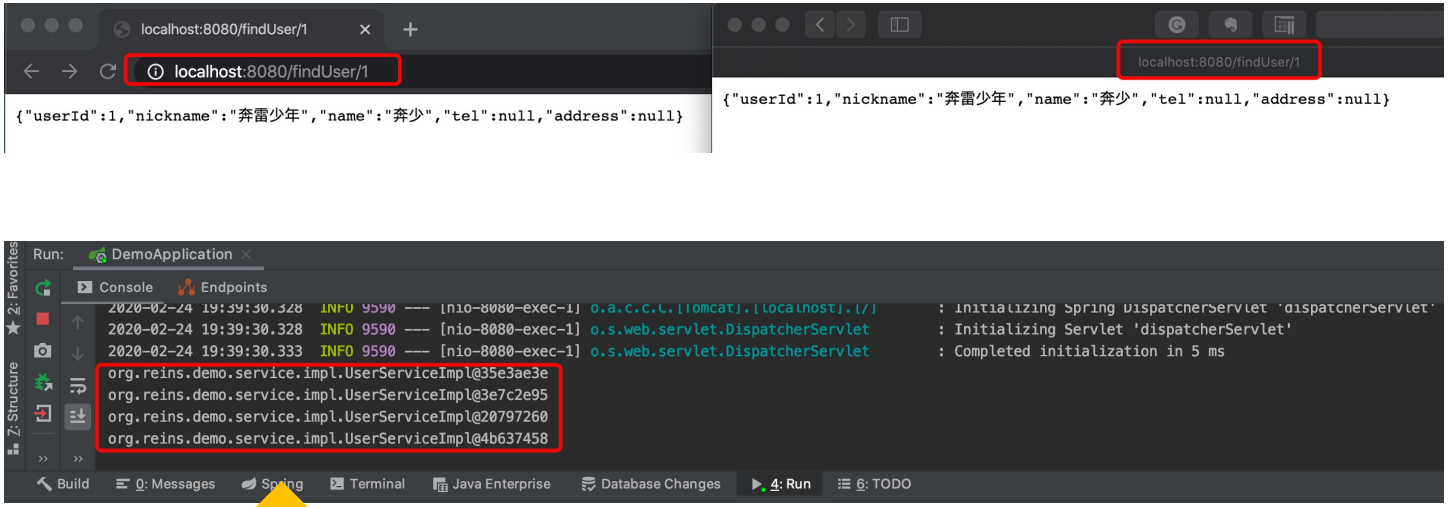
prototype



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

UserService

prototype

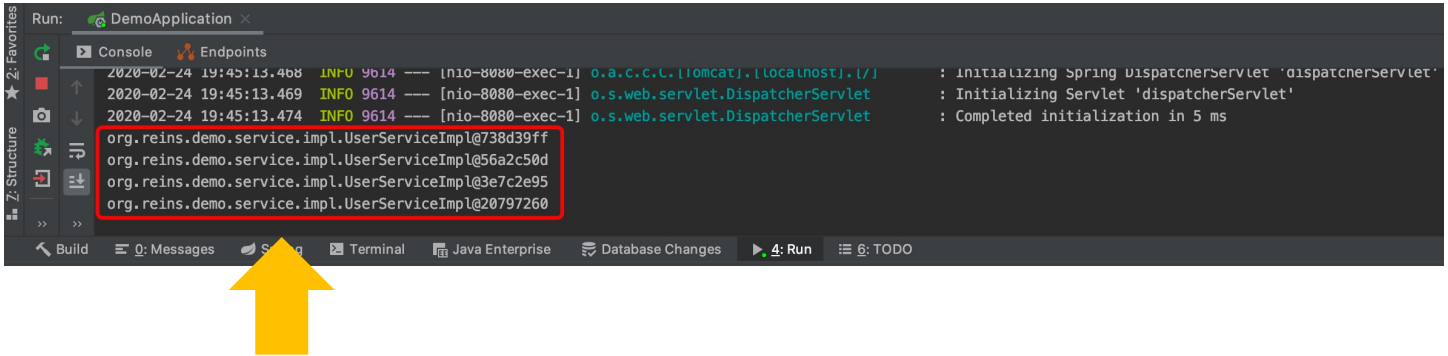
UserController.java

```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));
    };
}
```

UserServiceImpl.java

```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

UserService

session

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));
   };
}
```

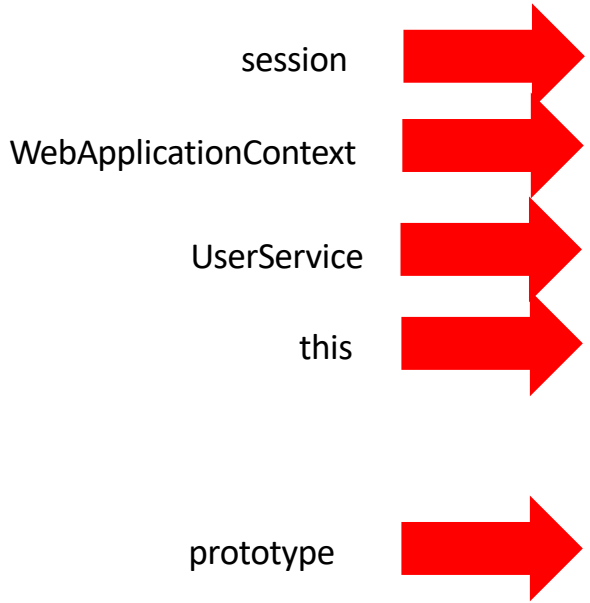
UserServiceImpl.java

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

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

Scope - session

Two Service instances

Scope - session



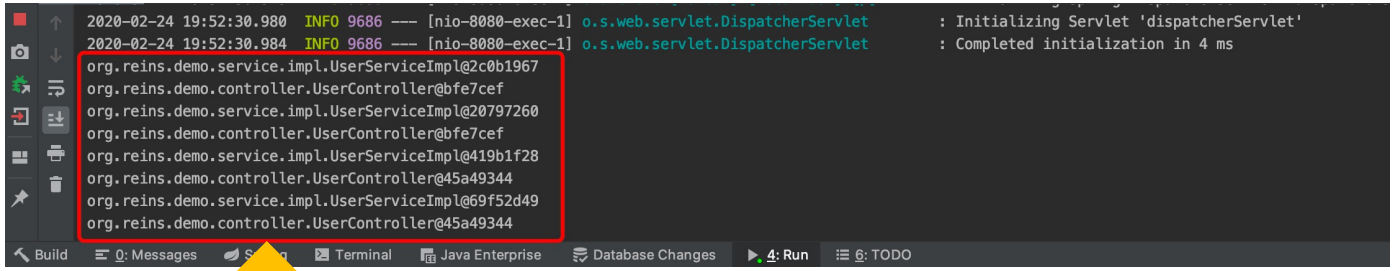
UserController.java

```java
@RestController
@Scope(”session")
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);
        System.out.println(this);
        return userService.findUserById(Integer.valueOf(id));
    };
}
```

UserServiceImpl.java

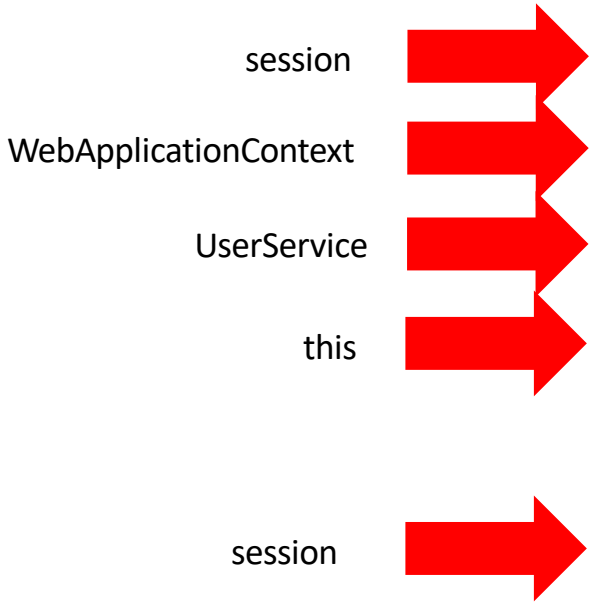
```java
@Service
@Scope(”prototype")
public class UserServiceImpl implements UserService {
    @Autowired
    private UserDao userDao;

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

Scope - session



Two Controller instances
Four Service instances

Scope - session



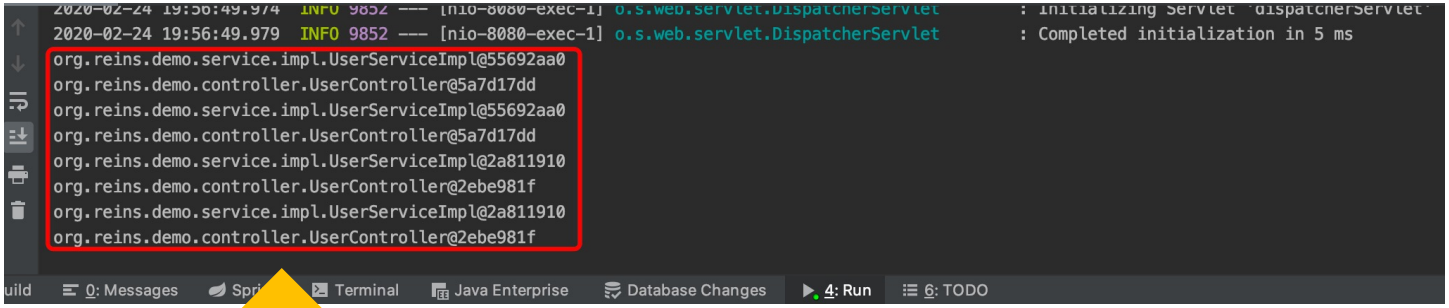
UserController.java

```
@RestController
@Scope(”session")
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);
      System.out.println(this);
      return userService.findUserById(Integer.valueOf(id));
   };
}
```

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



Two Controller instances
Two Service instances

Scope - session

prototype

WebApplicationContext

UserService

this

session

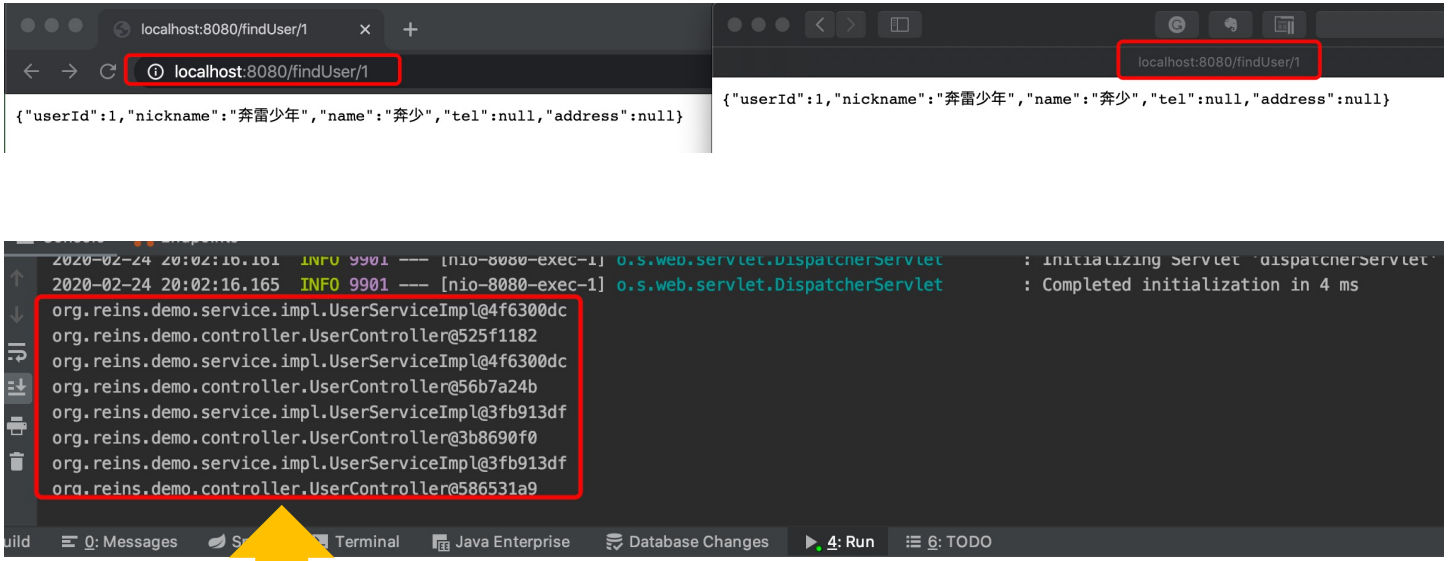
UserController.java

```
@RestController
@Scope(”prototype")
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);
      System.out.println(this);
      return userService.findUserById(Integer.valueOf(id));
   };
}
```

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



Four Controller instances
Two Service instances

Summary

- 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?

作业一(第一部分)

- 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）

References

- Web Applications: What are They? What of Them?
  - http://www.acunetix.com/websitesecurity/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!