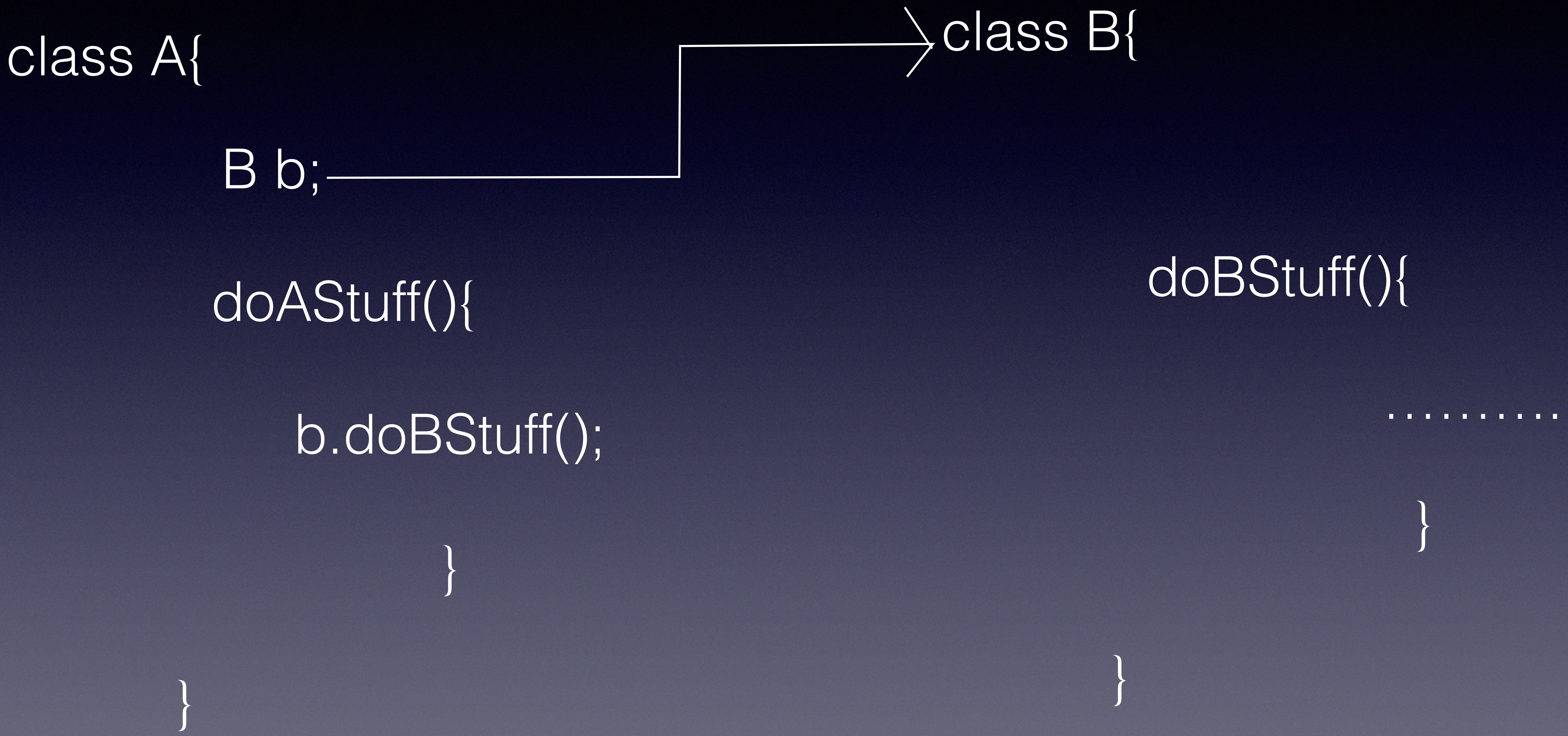
What is Spring

Dependency Injection

Compliments JEE

Dependency Injection

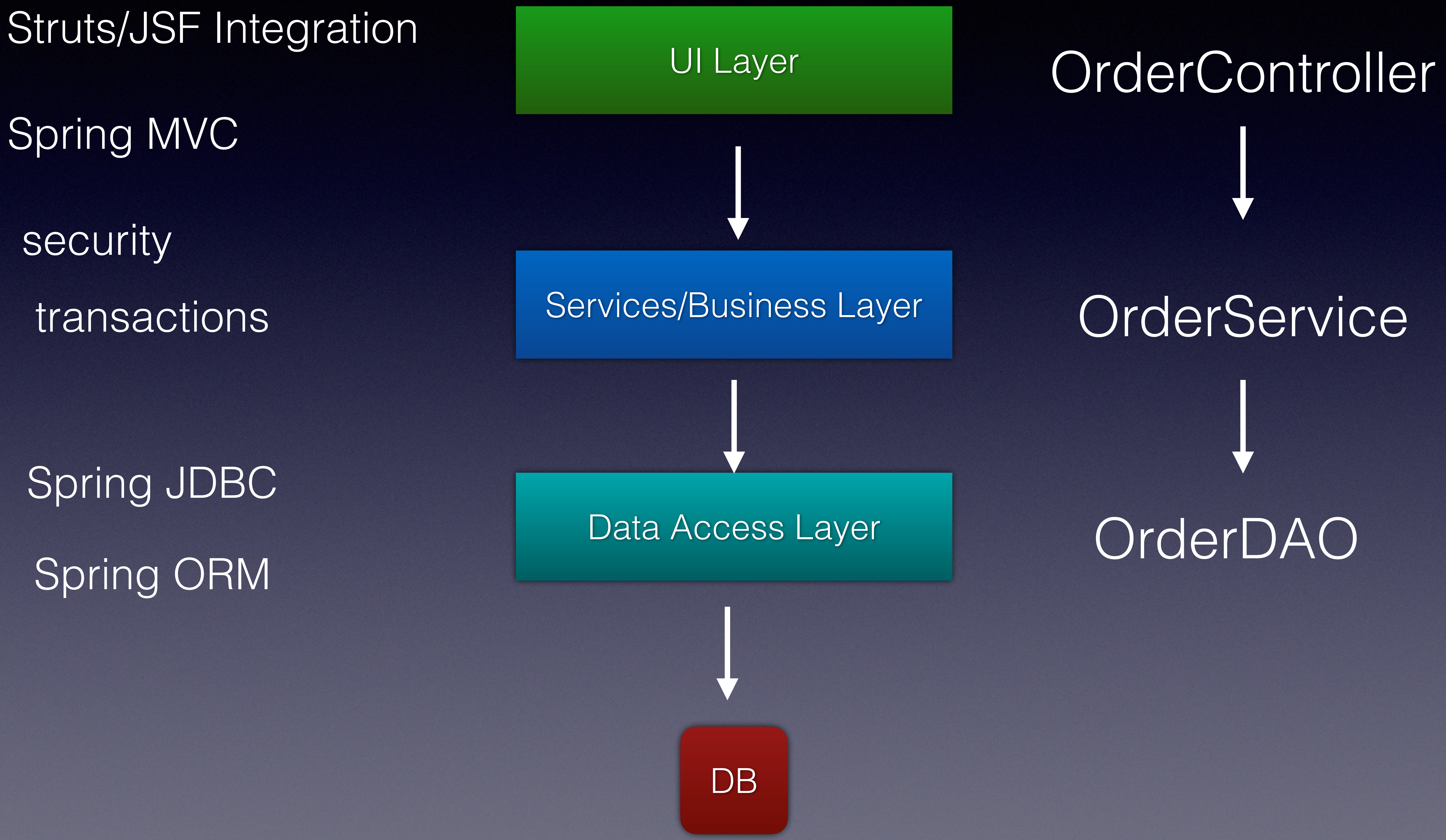


```
class A{

        B b;

        doAStuff(){

            b.doBStuff();

                }

}
```

```
class B{

        doBStuff(){

            ..........

            }

}
```

Inversion Of Control

Spring and JEE



Constructor Injection

```
<constructor-arg>
            <value> b </value>
<constructor-arg>

<constructor-arg>
            <ref bean="a"/>
</constructor-arg>
```

As Tag

As Attribute

C Shema/C Namespace

MIX

Primitive Types

```
<bean class="Product">
        <property name="id">
                <value>10</value>
```

value as element

value as attribute

p schema/p namespace

Collection Types

List

Set

Map

Properties

List

```
<bean>
    <property name="productIds">
        <list>
            <value>10 </value>
            <value> 20 </value>
             <null/>
```

Set

```
<bean>
    <property name=“productIds”>
            <set>
               <value>10 </value>
                <value> 20 </value>
```

Map

```
<bean>
    <property name="“productsInOrder”">
        <map>
            <entry key="100" value="IPhone" />
```

Properties

```
<bean>
        <property name=“languages”>
            <props>
                <prop key="USA">English</prop>
```

Reference Types

$$\begin{tikzcd} A \arrow[r, "\text{HAS - A}"] & B \end{tikzcd}$$

```
B b;

<bean class="B" name="b"/>

        <bean class="A" name="a">
            <property name="b">
                    <ref bean="b"/>
              </property>
         </bean>
```

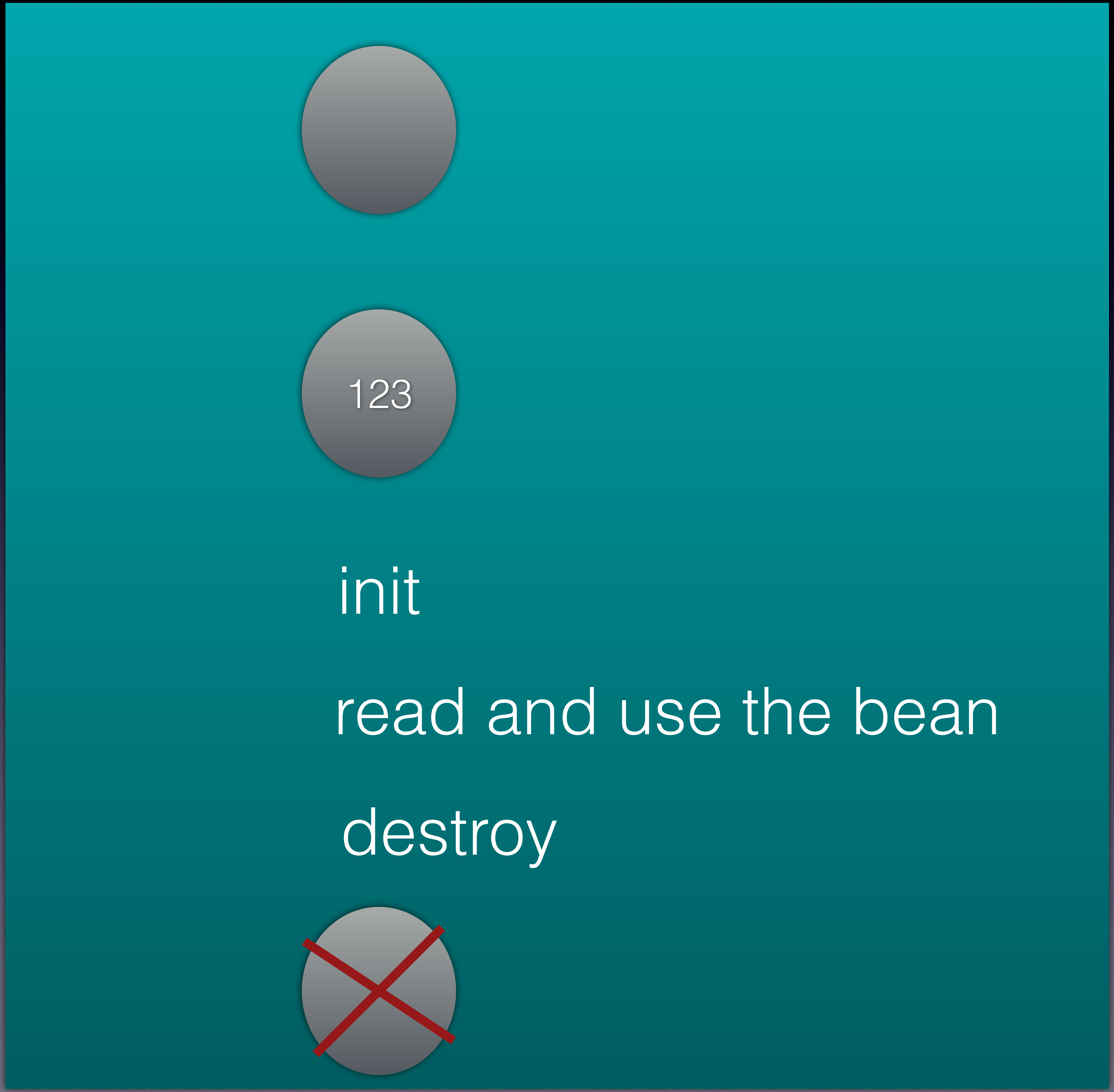
Life Cycle Methods

```
public void init()

public void destroy()
```

Spring Container:

Spring Bean (.java)

Config File (.xml)



1) XML Configuration

2)Spring Interfaces

3)Annotations

Dependency Check

Product

id

name

Connection

username

password

dburl

Container

2.x<=	XML Configuration
3.0	@Required

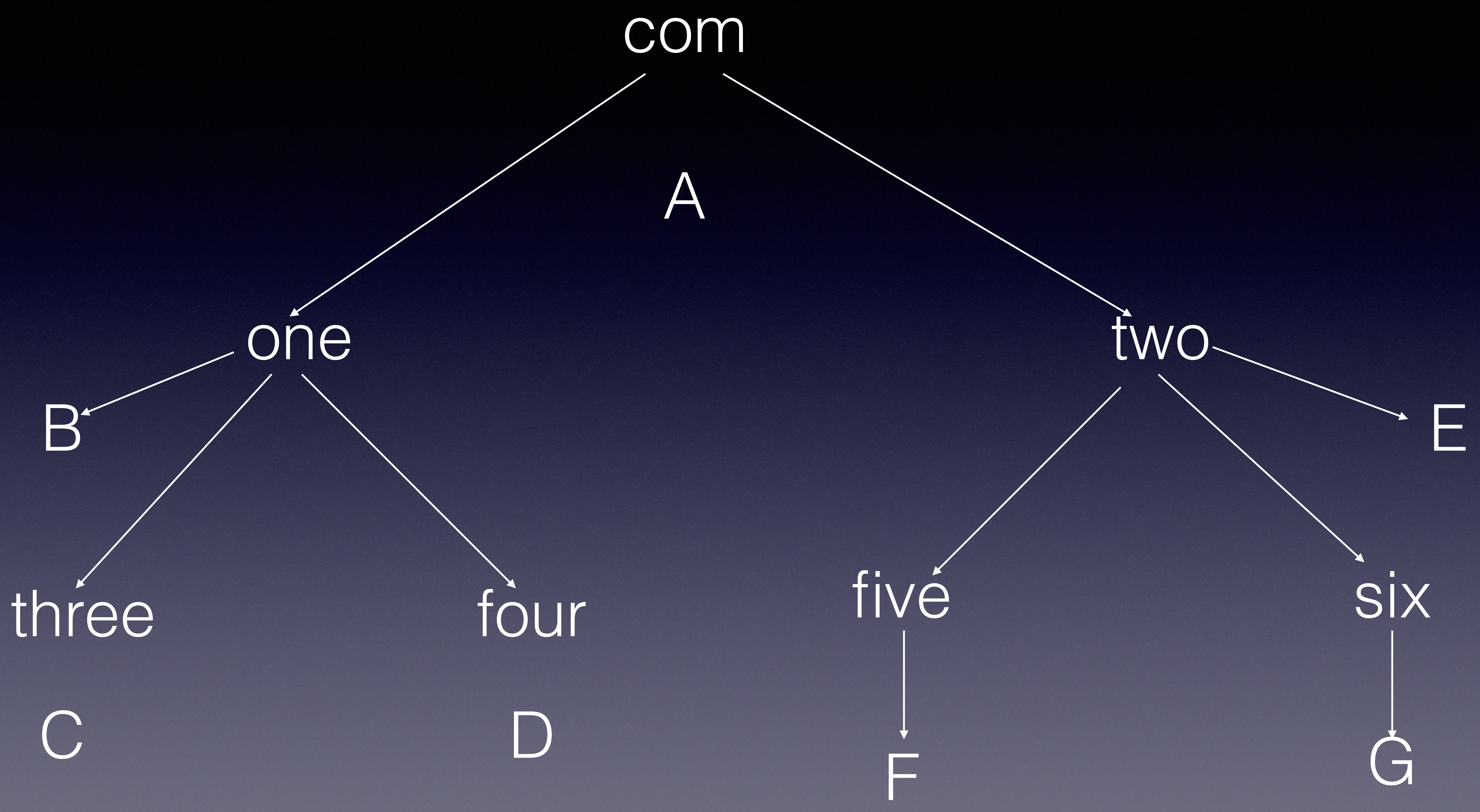
Stereotype Annotations

XML <bean…>

@Component

class Instructor

<context:component-scan base-package="com.bharath"/>


com
A
one
two
B
E
three
four
five
six
C
D
F
G

```
@Component
    class Instructor
```

```
Instructor instructor = new Instructor();
```

User Defined Types

SpEL Spring Expression Language

Expression

@Value

Classes,Variable,Methods,Constructors and Objects

and symbols

char,numerics,operators,keywords and special symbols which return a value

```
@value("#{66+44}")
@value("#{5>6?22:33"})
```

static methods

object methods

variables

T(class).method(param)

singleton

prototype

@Scope request

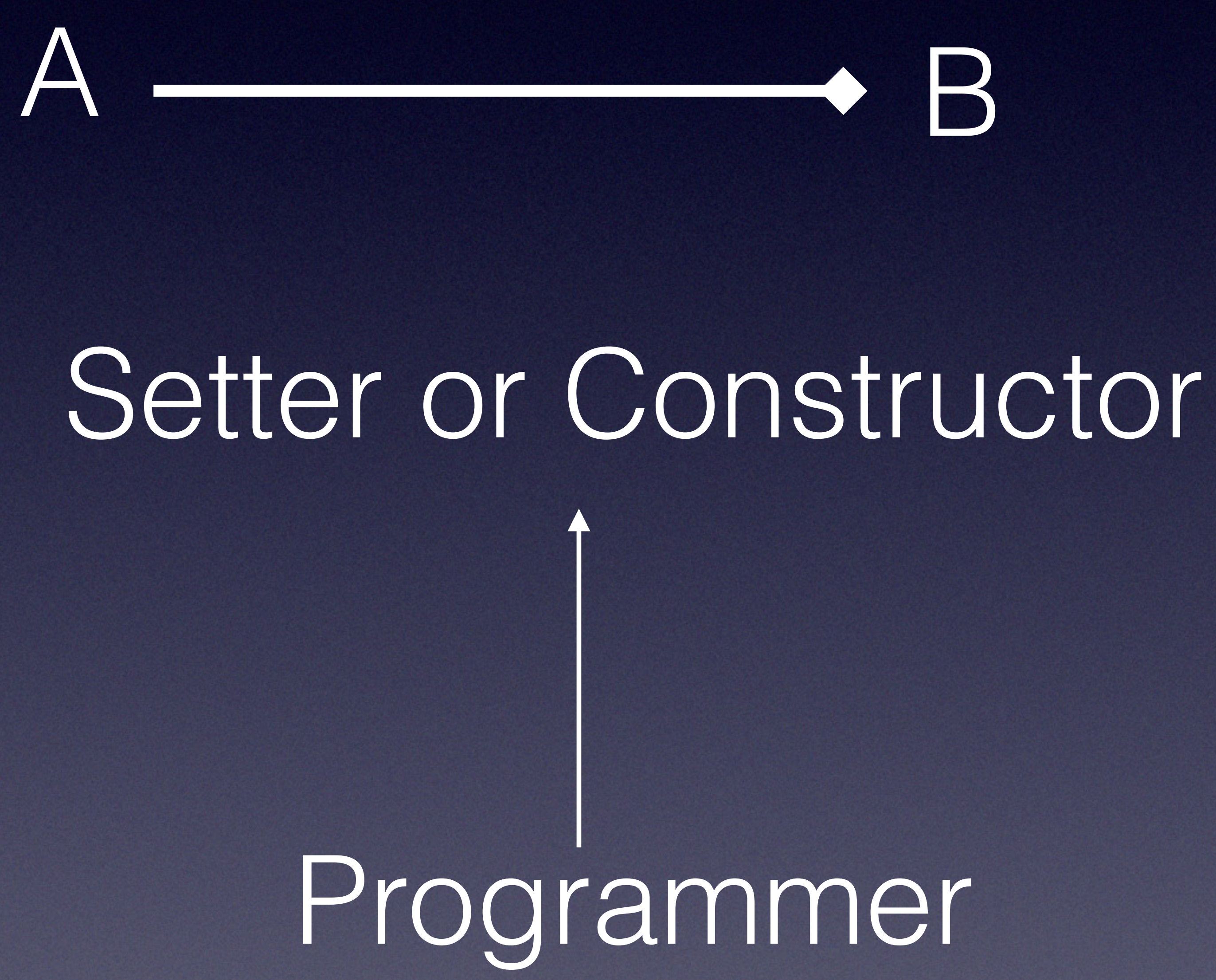
session

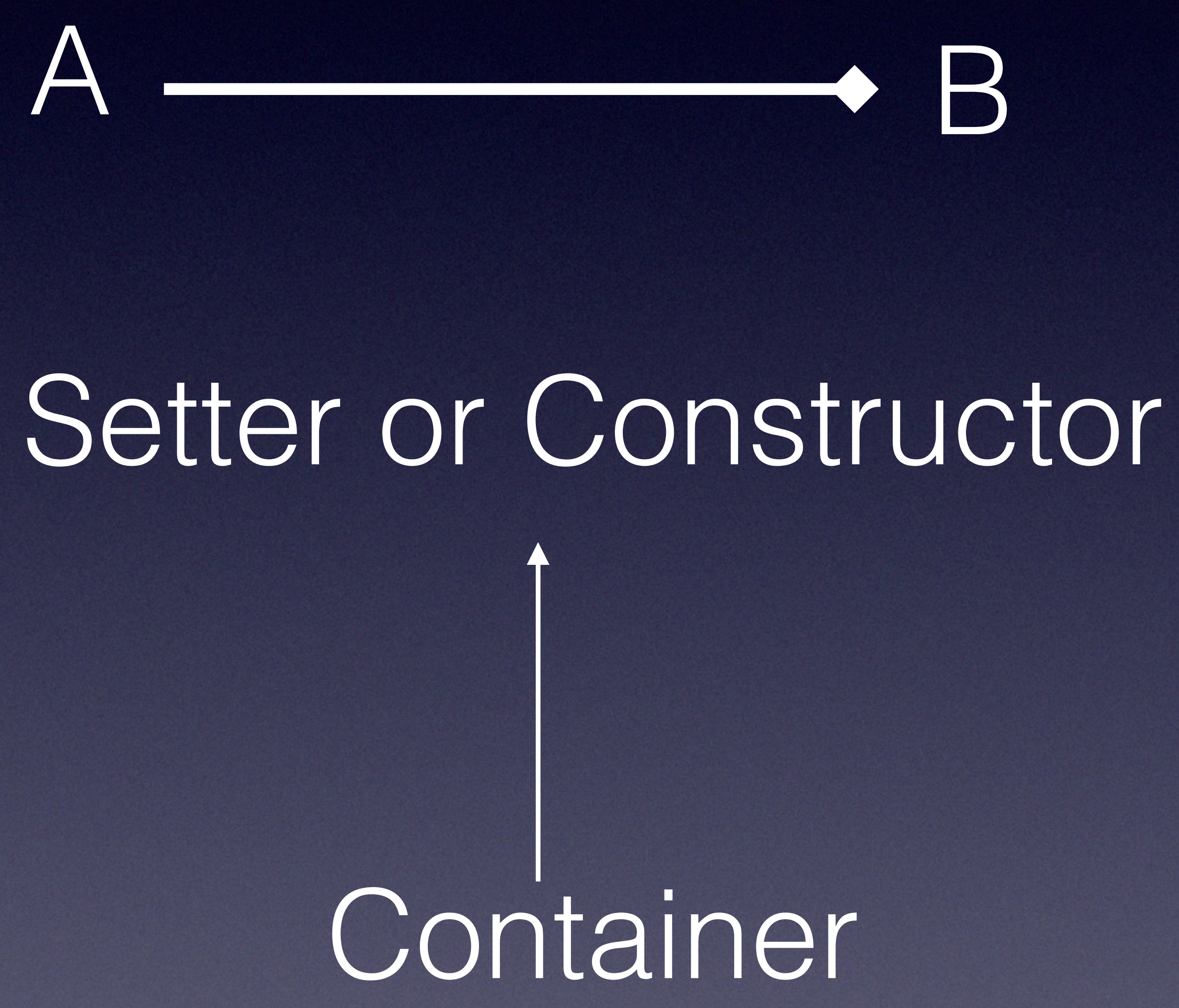
globalsession

@Value

Primitive Types	@Value(“20”)
	@Value(“Core Java”)
Collection Types	util:Cn id=“myList”
	@Value(“#{myList}”)
Object Types	@Autowired

Wiring and Auto Wiring





Autowiring

XML

- No (Default)
- By Type
- By Name
- Auto Detect
- By Constructor

Annotations

- @Autowired
- @Qualifier

Bean Externalization or Reading Properties

database.properties

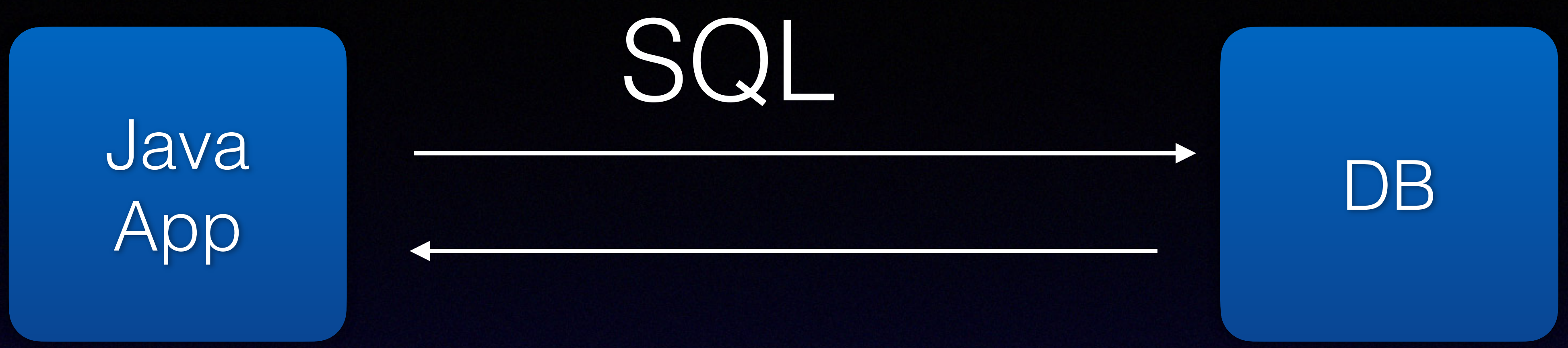
- dbName
- port
- userName
- password

property place holder configuration

- Create the properties
- Link the properties
- User properties in xml and inject

Spring JDBC

JDBC



Connection con = DriverManager.getConnection….

Statement statement = con.createStatement…

Spring JDBC

JDBCTemplate

JDBC + Template

Technology Design Pattern

Template Common Code

Developer loves JDBCTemplate

$$\begin{tikzcd}
\text{JDBCTemplate} \arrow[r] & \text{javax.sql.DataSource(I)} \\
 & \text{DriverManagerDataSource} \arrow[u]
\end{tikzcd}$$

driverClassName

url

userName

password

JDBCTemplate

update(String sql) int

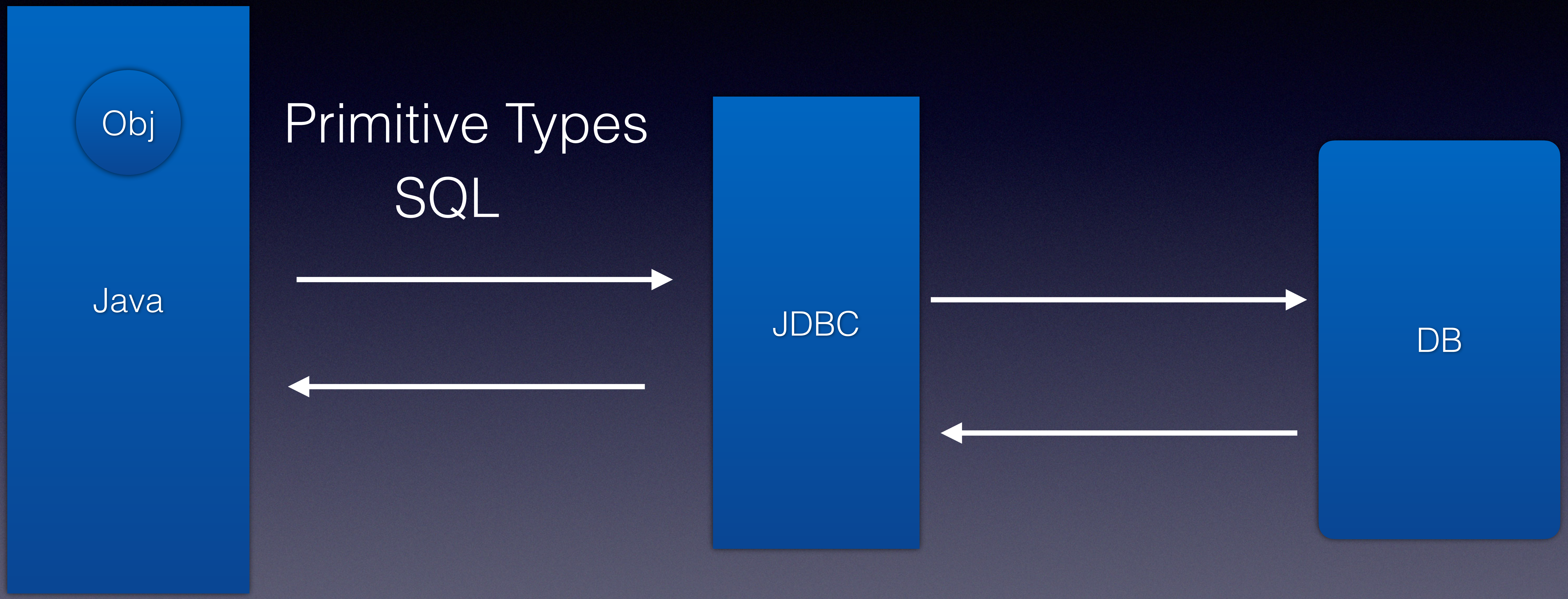
update(String sql,Object…args) int

insert,update and delete

DriverManagerDataSource		dataSource
	driverClassName	com.mysql.jdbc.Driver
	url	jdbc:mysql://localhost/mydb
	username	root
	password	test
JDBCTemplate		jdbcTemplate
	dataSource	

Create the test class and use the JDBCTemplate

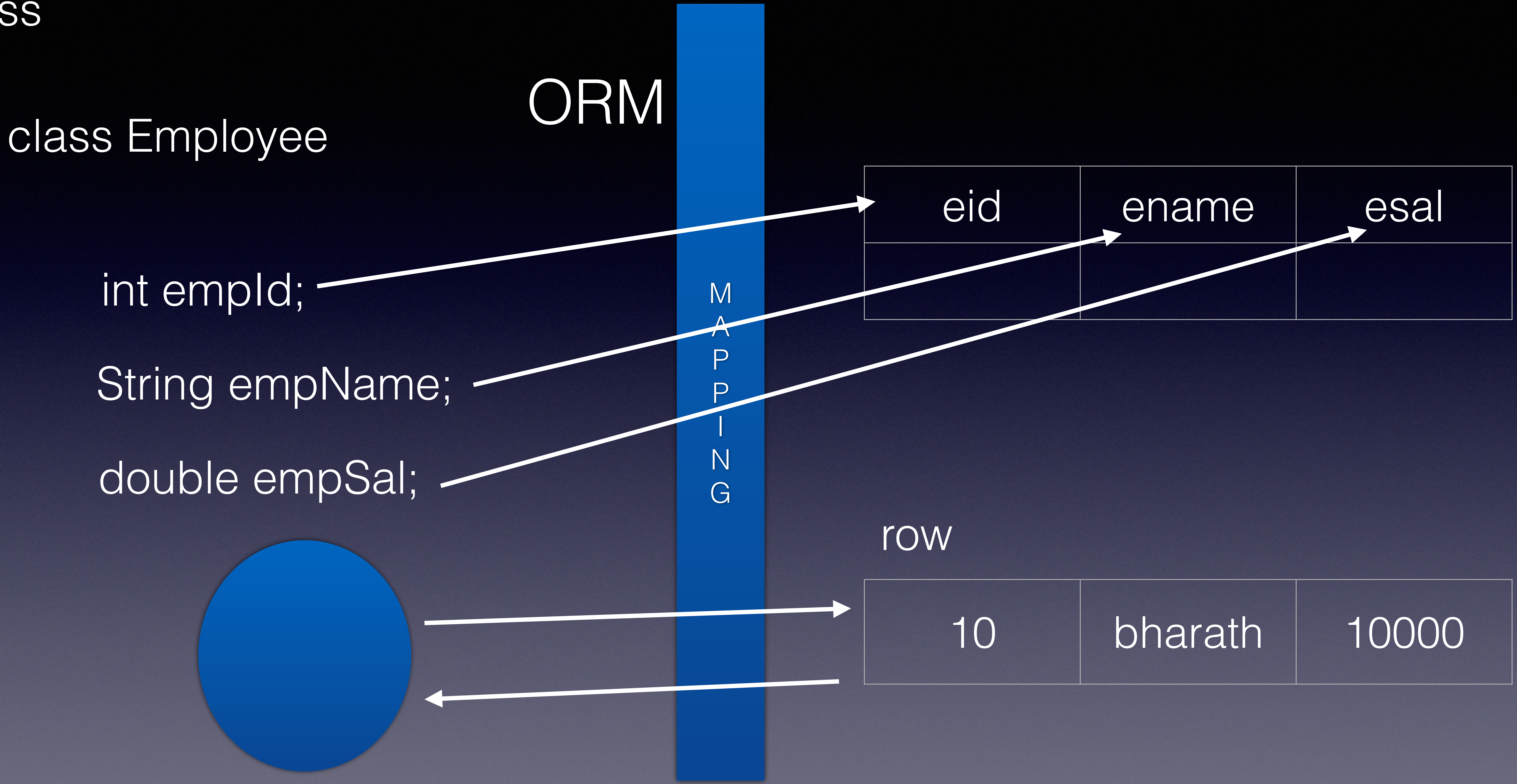
Spring ORM



Mapping:

```
@Entity
@Table(name="product")
 class Product{

        @Id
        @Column
         int id;

        @Column
        String name;
```

Class


ORM
class Employee
int empId;
String empName;
double empSal;
MAPPING
eid
ename
esal
row
10
bharath
10000

Product

$$\begin{tikzcd}
\text{ProductDao} \arrow[r] & \text{HibernateTemplate} \arrow[r] & \text{SessionFactory} \\
\text{ProductDaoImpl} \arrow[u] & & \text{LocalSessionFactoryBean} \arrow[u] \\
\text{DataSource} & & \text{TransactionManager}
\end{tikzcd}$$

AnnotationSessionFactoryBean

dataSource

hibernateProperties

annotatedClasses

ORM Mapping:

XML and Annotations

@Entity

@Table

@Id

@Column

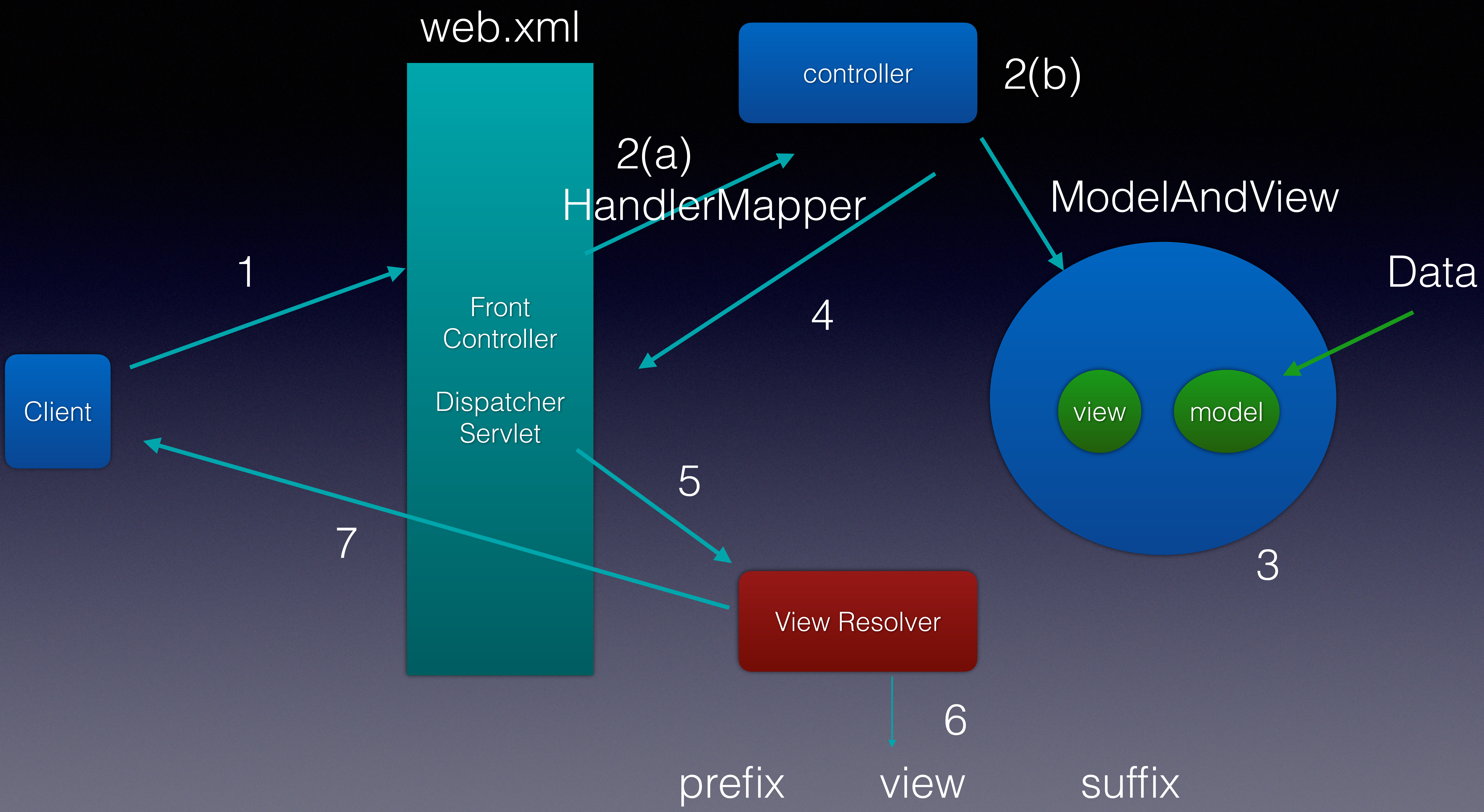
```
@Entity

@Table

public class Employee{

        @Id
        @Column(name="id")
        private int id;
        @Column(name="firstname")
        private int firstName;
        @Column(name="lastName")
        private int lastName;
```

Spring MVC

Front Controller

Handler Mapper

View Resolver


web.xml
controller
2(b)
2(a)
HandlerMapper
ModelAndView
1
Data
4
Front Controller
Dispatcher Servlet
Client
view
model
5
7
3
View Resolver
6
prefix
view
suffix

ViewResolver

prefix	view	suffix
views/	hello	.jsp

Spring MVC Application Creation Steps:

Configure the dispatcher servlet

Create the spring configuration

Configure the View Resolver

Create the controller

Create the folder structure and view

Sending Data

Controller to the UI

UI to the Controller

Controller to the UI

ModelAndView

$\text{addObject}(\underset{\overset{\uparrow}{\text{String}}}{\text{key}},\underset{\overset{\uparrow}{\text{Object}}}{\text{value}})$

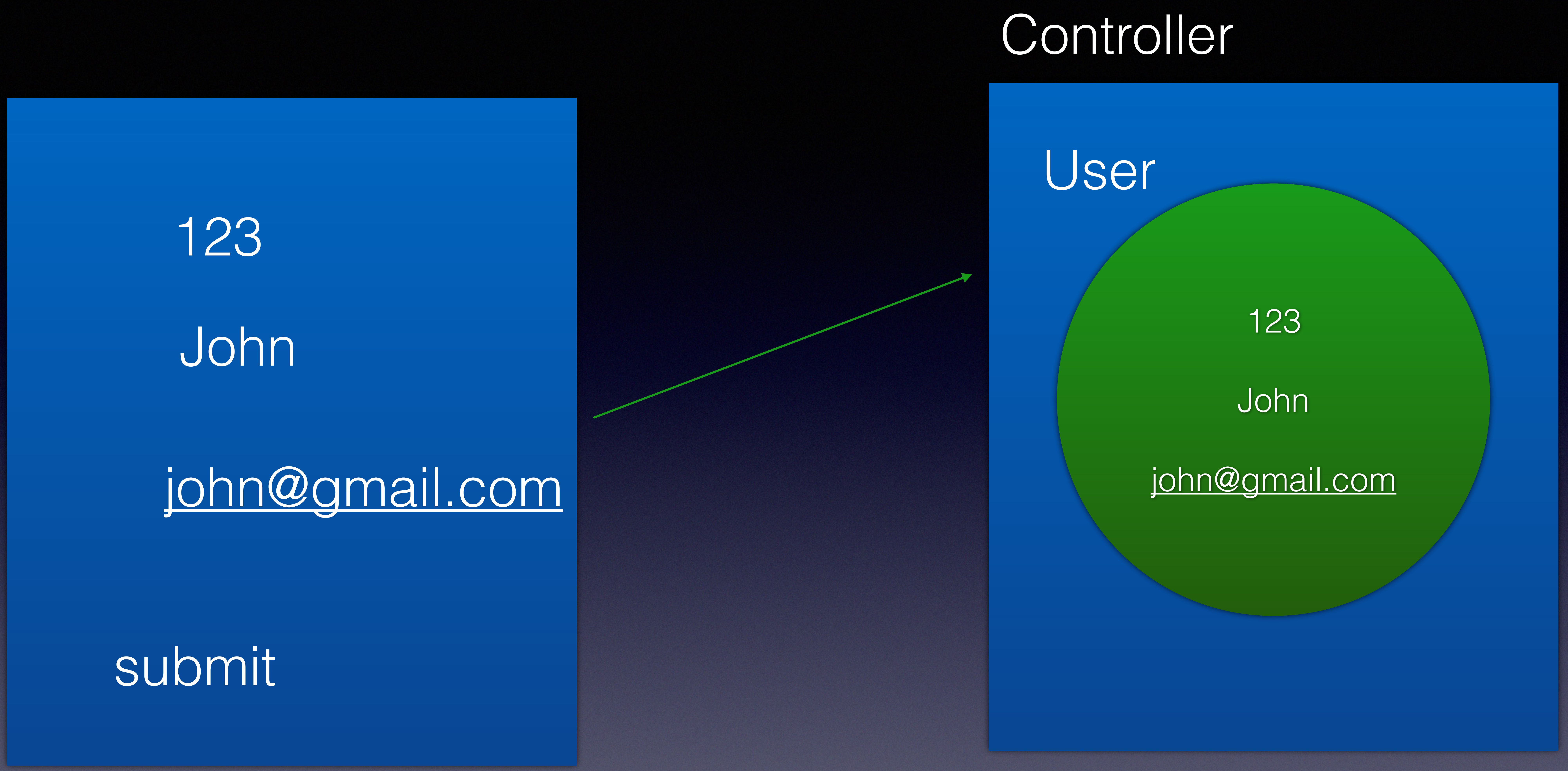
request.getAttribute(“key”)

Primitive Object Collection

Sending data from UI to Controller:

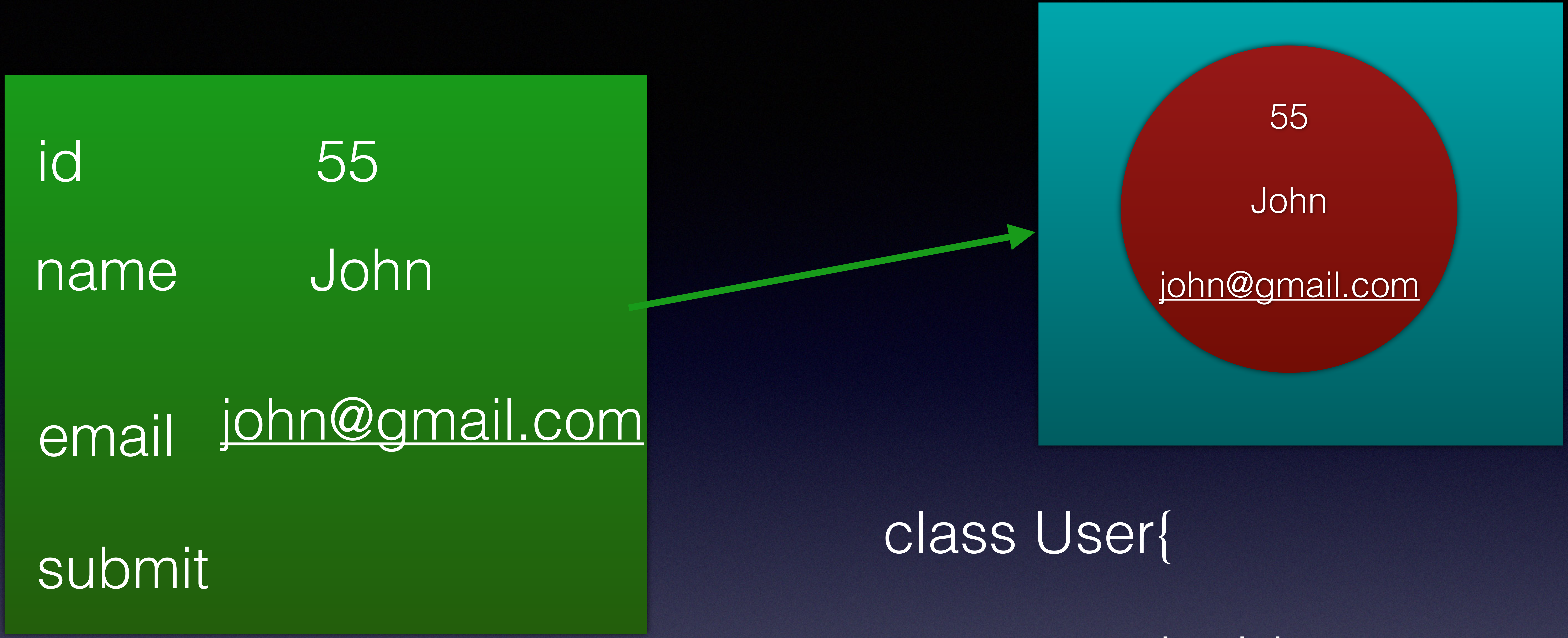
HTML Form

Query Parameters



Container:

Reads Converts Creates Object Sets Values

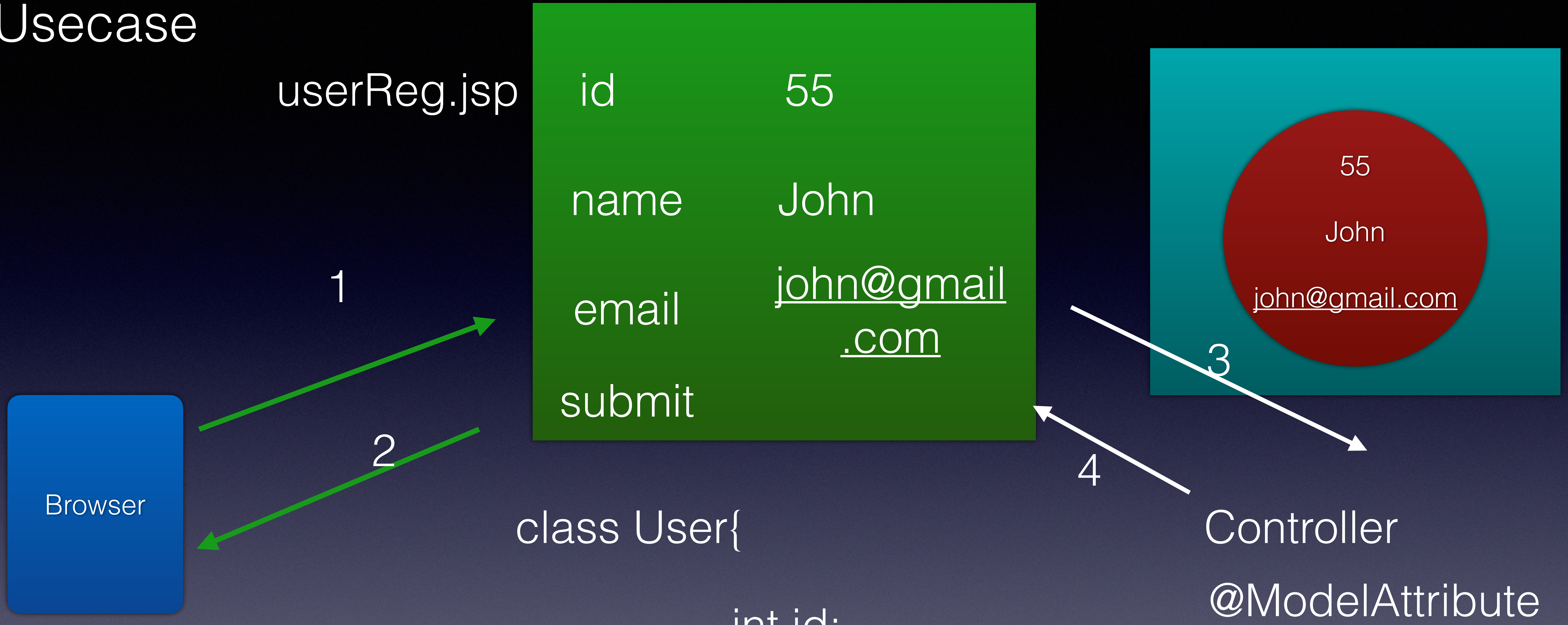


Controller

@ModelAttribute

```
class User{
        int id;
        String name;
        String email;
        }
```

Usecase



```
class User{
        int id;
        String name;
        String email;
}
```

Coding Steps

Model
Controller
Services
DAL
UserController
UserService
UserDao
HibernateTemplate
UserServiceImpl
UserDaoImpl

Create the Maven Project

web.xml DispatcherServlet

HibernateTemplate

SessionFactory

dispatcher-servlet.xml

DataSource

ViewResolver

Model

DAL Layer

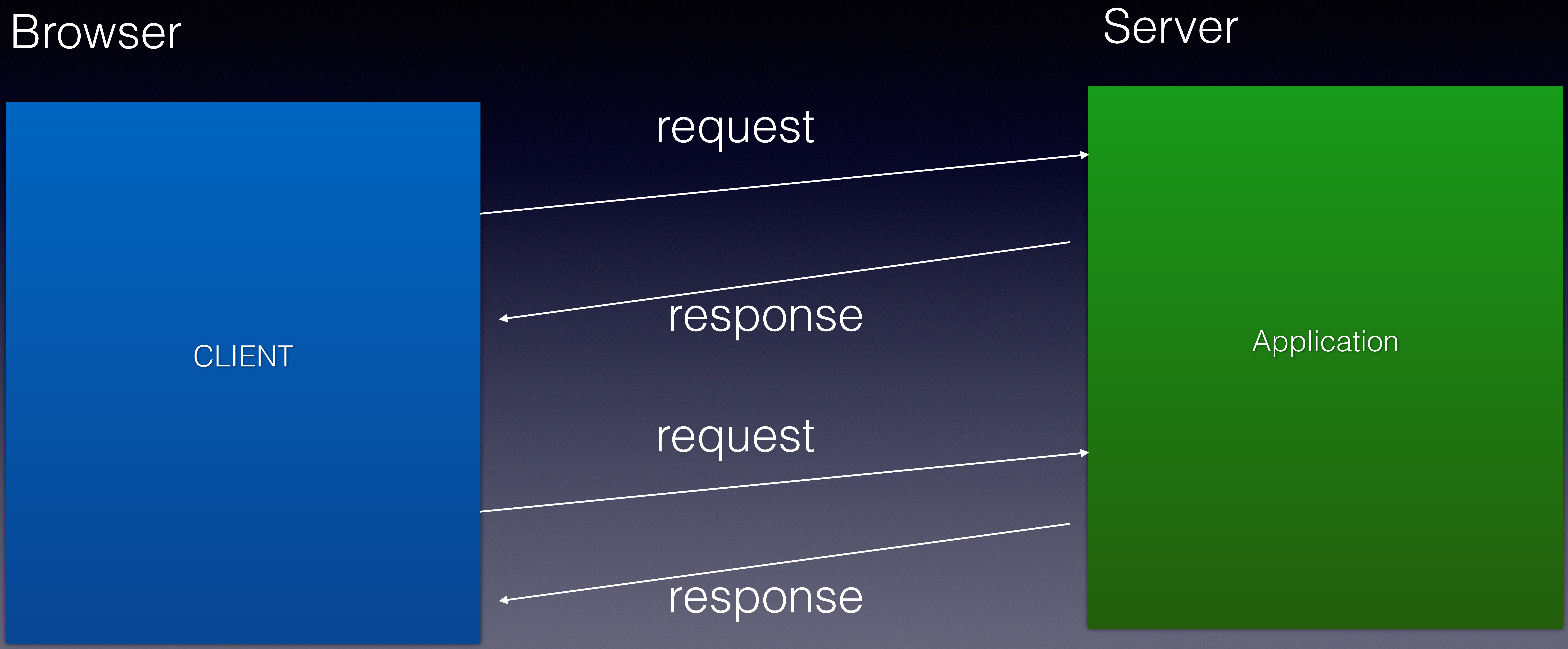
Code

Services Layer

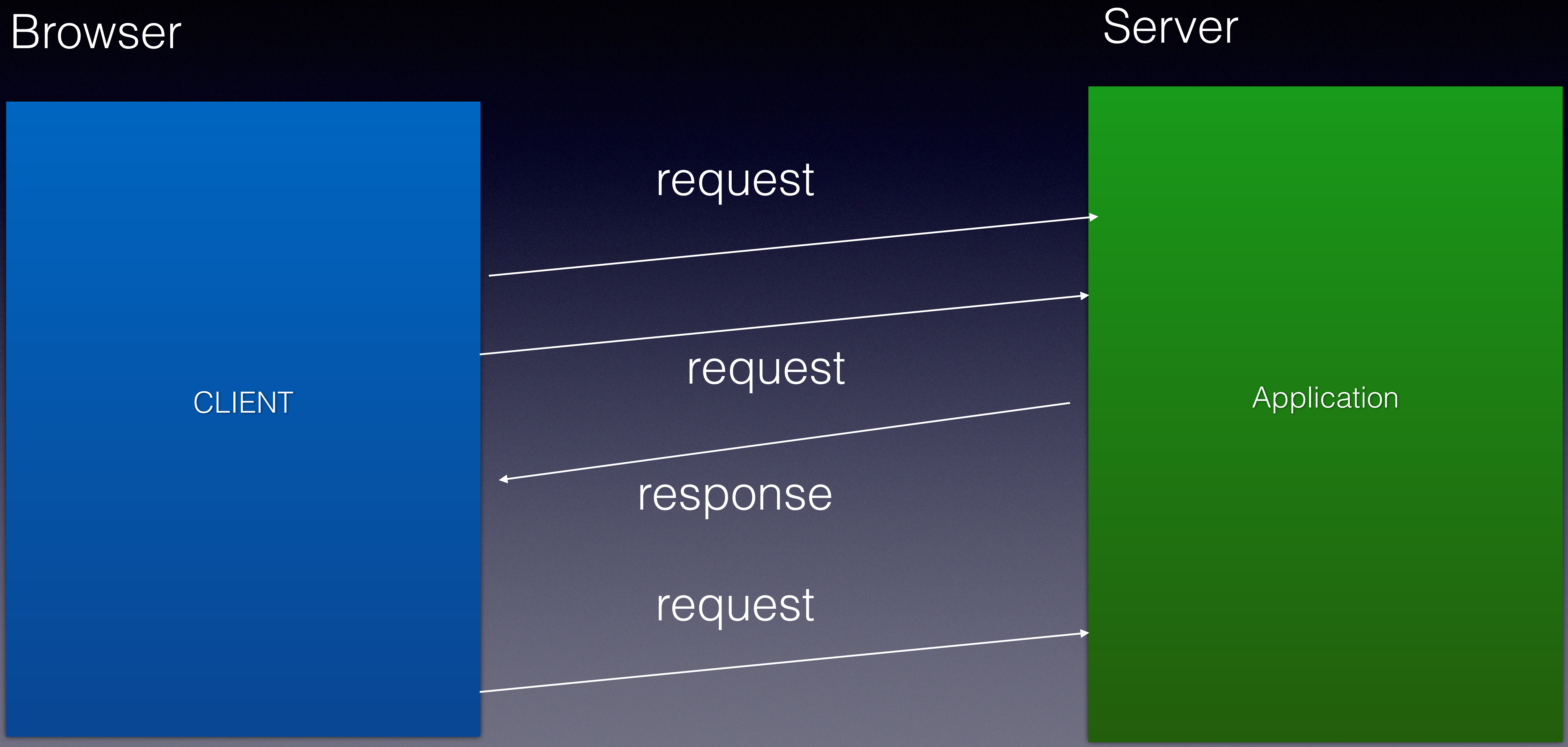
Controller Layer

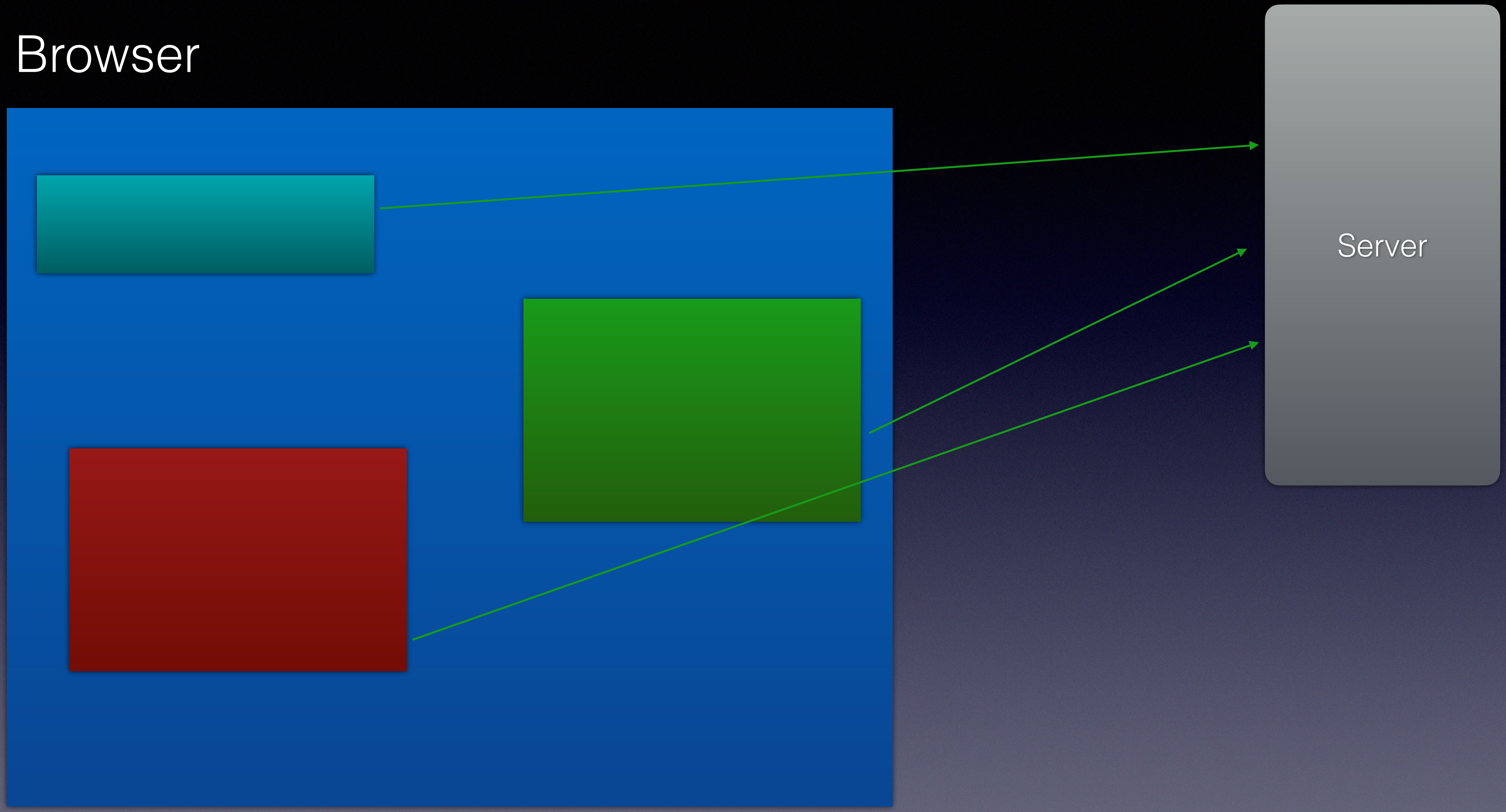
AJAX

Synchronous



Asynchronous




Browser
Server

JQuery

Java Script

$

${“#userId”}

```
$.ajax({

        url:””

        data:{key:value}

        success:function(){}
```

UseCase Steps

Implement the backend validation

$$\begin{tikzcd}\text{Controller} \arrow[r] & \text{Service} \arrow[r] & \text{DAO}\end{tikzcd}$$

Make the AJAX Call

Use JQuery

onChange

AJAX Call

Controller

Handle Response