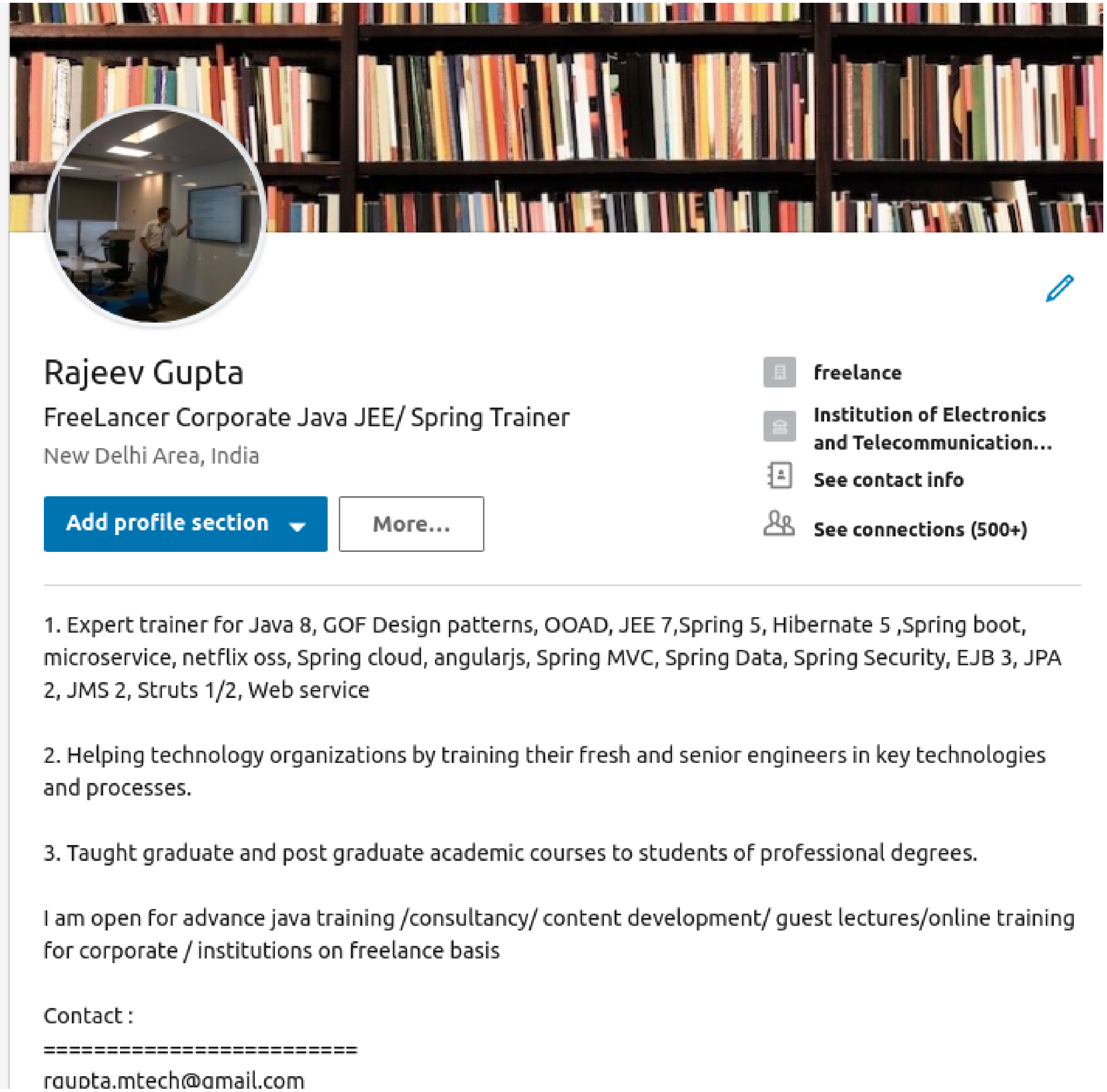
Spring with spring boot...

Rajeev Gupta
rgupta.mtech@gmail.com
Java Trainer & consultant



Rajeev Gupta

FreeLancer Corporate Java JEE/ Spring Trainer

New Delhi Area, India

1. Expert trainer for Java 8, GOF Design patterns, OOAD, JEE 7,Spring 5, Hibernate 5 ,Spring boot, microservice, netflix oss, Spring cloud, angularjs, Spring MVC, Spring Data, Spring Security, EJB 3, JPA 2, JMS 2, Struts 1/2, Web service

2. Helping technology organizations by training their fresh and senior engineers in key technologies and processes.

3. Taught graduate and post graduate academic courses to students of professional degrees.

I am open for advance java training /consultancy/ content development/ guest lectures/online training for corporate / institutions on freelance basis

Contact :
=========================
rgupta.mtech@gmail.com

Clients:
================
Gemalto, Noida
Cyient Ltd, Noida
Fidelity Investment Ltd
Blackrock, Gurgaon
Mahindra comviva
Steria
Bank Of America
incedo gurgaon
MakeMyTrip
Capgemini India
HCL Technologies
CenturyLink
Deloitte consulting
Nucleus Software
Ericsson Gurgaon
Avaya gurgaon
Kronos Noida
NEC Technologies
A.T. Kearney
ust global
TCS
North Shore Technologies Noida

IBM
Sapient
Accenture
Incedo
Genpact
Indian Air force
Indian railways
Vidya Knowledge Park

Session 1

- Introduction Spring framework, where it fits, design patterns
- Dependency Injection, Configuration-XML, Java
- Spring Boot what why?
- Aspect Oriented Programming, how it helps, configuration, examples
- Spring Hibernate

Session 2

- Spring REST
- Spring MVC
- Spring security

Session 1

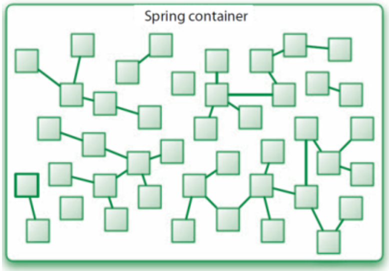
- **Introduction Spring framework, where it fits, design patterns**
- Dependency Injection, Configuration-XML, Java
- Spring Boot what why?
- Aspect Oriented Programming, how it helps, configuration, examples
- Spring Hibernate

What is spring framework?

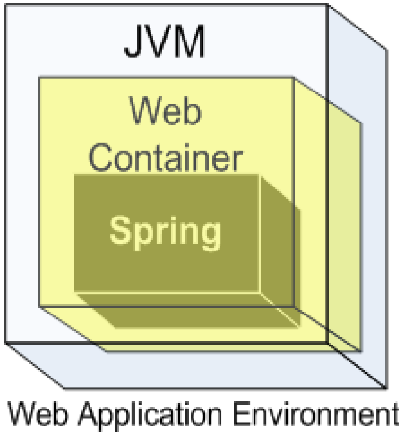
Spring framework is a contrainer that manage life cycle of bean.

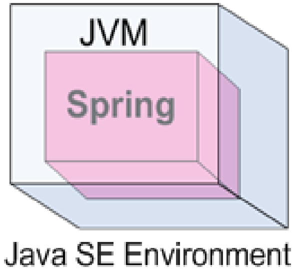
It Does 2 primary Jobs:

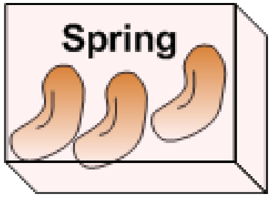
1. bean wiring
2. bean weaving

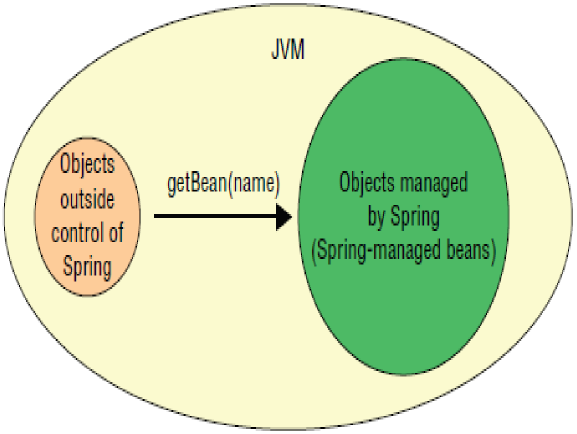


A bean is an object that is instantiated, assembled, and managed by a Spring IoC container.





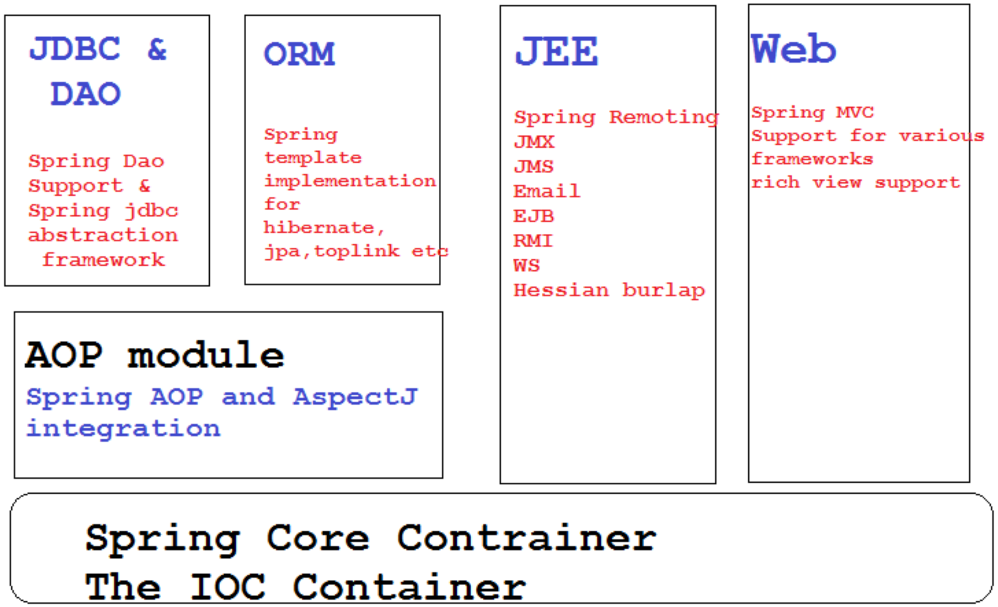



JVM
Objects outside control of Spring
getBean(name)
Objects managed by Spring (Spring-managed beans)

Why Spring framework? Where it fits?

Spring Framework is focused on simplifying enterprise Java development through

- dependency injection
- aspect-oriented programming
- boiler-plate code reduction using template design pattern



JDBC & DAO

Spring Dao Support & Spring jdbc abstraction framework

ORM

Spring template implementation for hibernate, jpa,toplink etc

JEE

Spring Remoting
JMX
JMS
Email
EJB
RMI
WS
Hessian burlap

Web

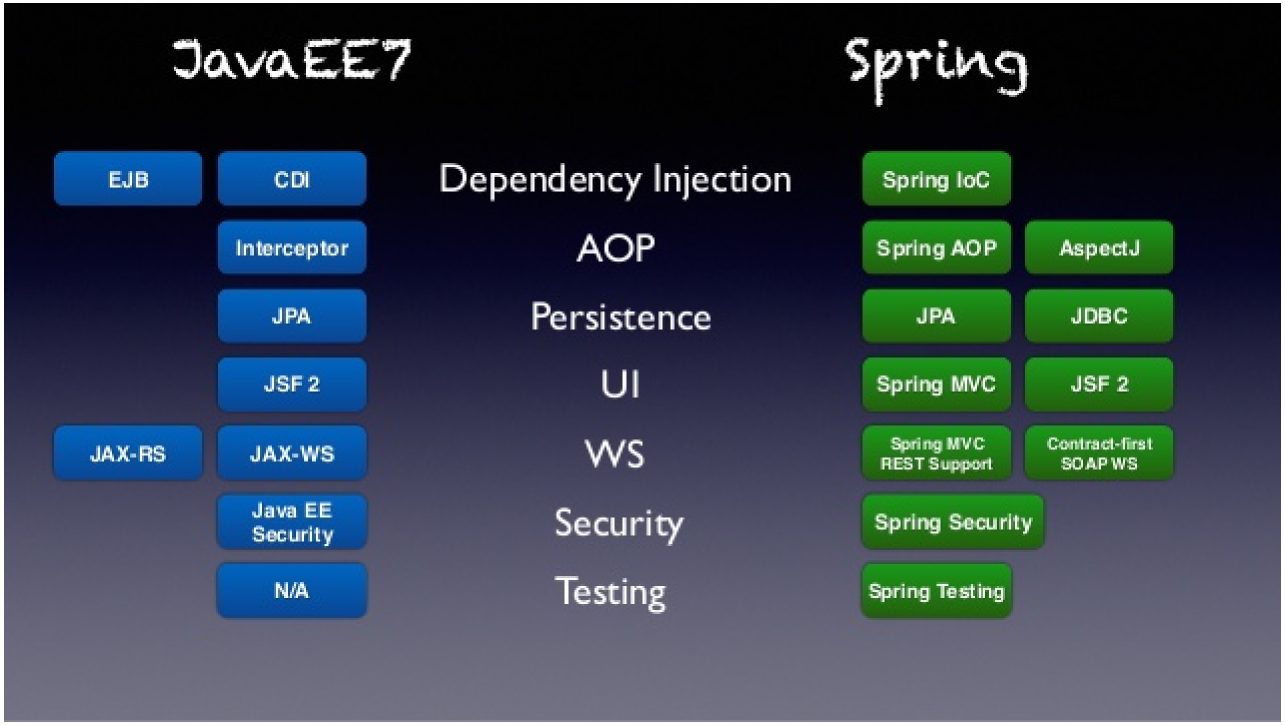
Spring MVC
Support for various frameworks
rich view support

AOP module

Spring AOP and AspectJ integration

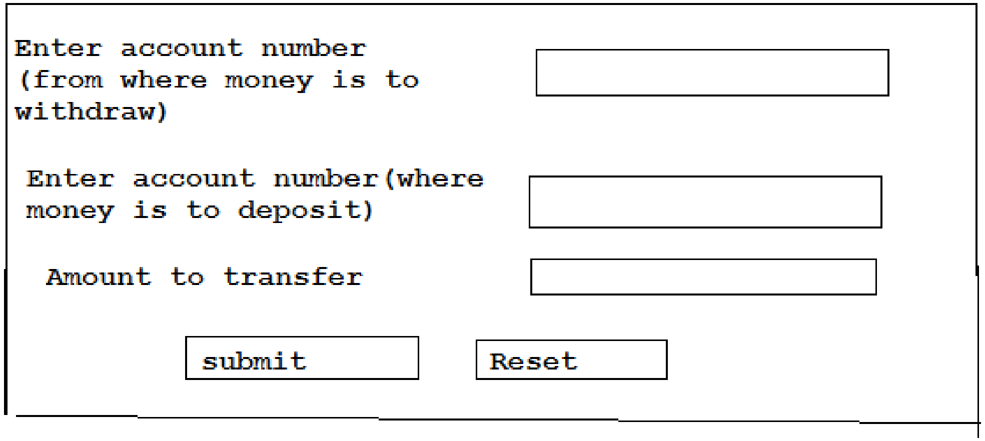
Spring Core Contrainer
The IOC Container

Java EE vs Spring framework



JavaEE7			Spring	
EJB	CDI	Dependency Injection	Spring IoC	
	Interceptor	AOP	Spring AOP	AspectJ
	JPA	Persistence	JPA	JDBC
	JSF 2	UI	Spring MVC	JSF 2
JAX-RS	JAX-WS	WS	Spring MVC REST Support	Contract-first SOAP WS
	Java EE Security	Security	Spring Security	
	N/A	Testing	Spring Testing	

Spring, Hibernate 3 Tier architecture



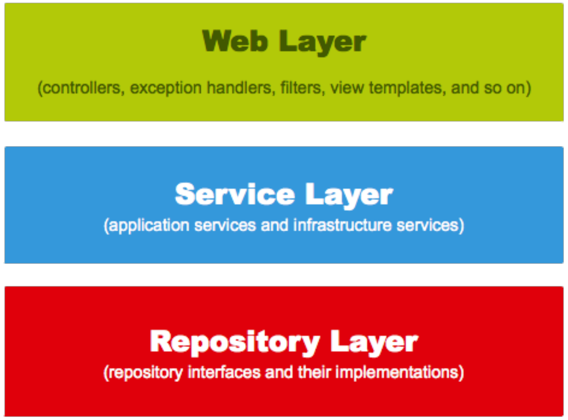
Enter account number (from where money is to withdraw)

Enter account number(where money is to deposit)

Amount to transfer

submit

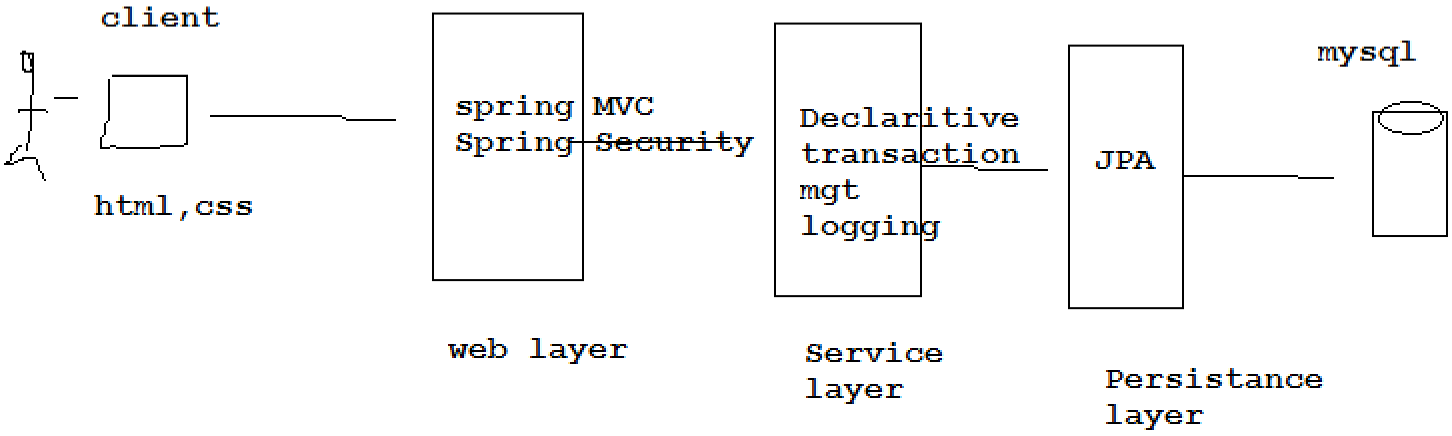
Reset



Web Layer
(controllers, exception handlers, filters, view templates, and so on)

Service Layer
(application services and infrastructure services)

Repository Layer
(repository interfaces and their implementations)



client

html,css

spring MVC
Spring ~~Security~~

web layer

Declaritive transaction mgt
logging

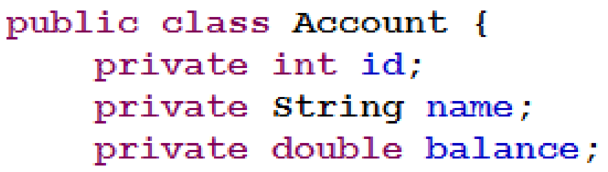
Service layer

JPA

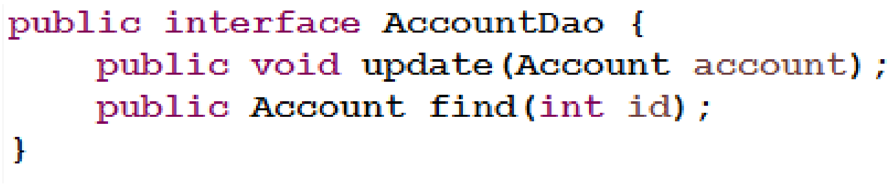
Persistance layer

mysql

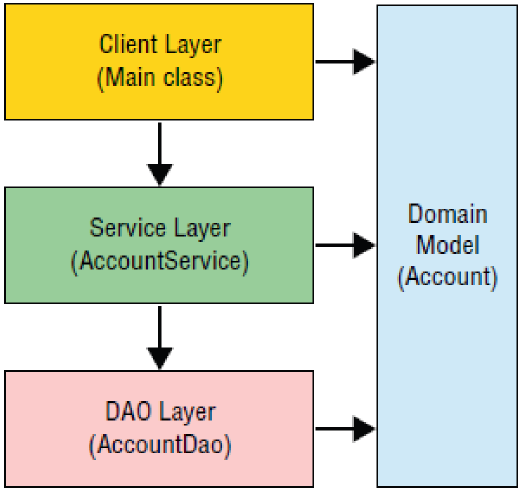
Application example: Fund transfer Bank Application

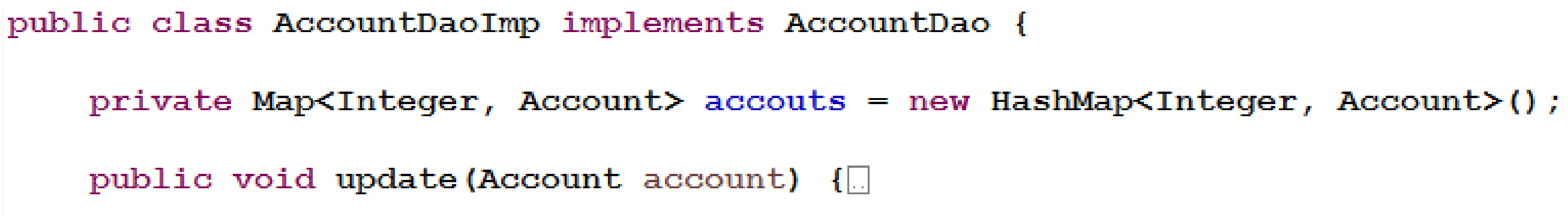


```java
public class Account {
    private int id;
    private String name;
    private double balance;
```

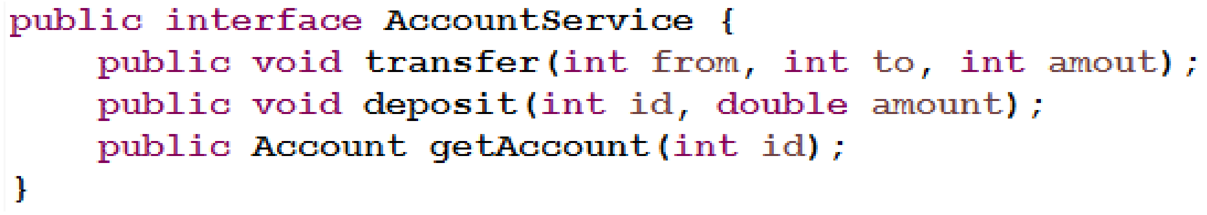


```java
public interface AccountDao {
    public void update(Account account);
    public Account find(int id);
}
```





```java
public class AccountDaoImp implements AccountDao {

    private Map<Integer, Account> accouts = new HashMap<Integer, Account>();

    public void update(Account account) {..
```



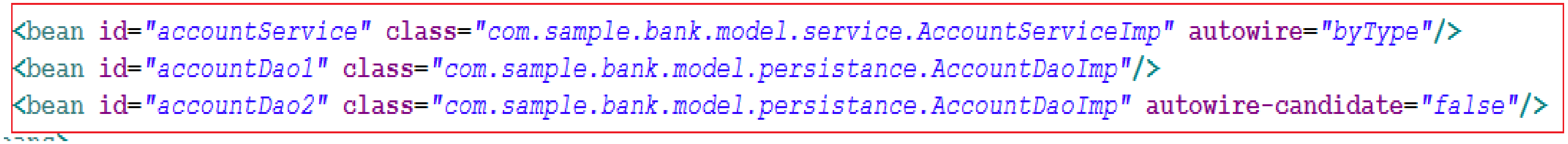
```java
public interface AccountService {
    public void transfer(int from, int to, int amout);
    public void deposit(int id, double amount);
    public Account getAccount(int id);
}
```

Session 1

- Introduction Spring framework, where it fits, Spring boot design patterns
- **Dependency Injection, Configuration-XML, Java**
- Spring Boot What, Why?
- Aspect Oriented Programming, how it helps, configuration, examples
- Spring Hibernate

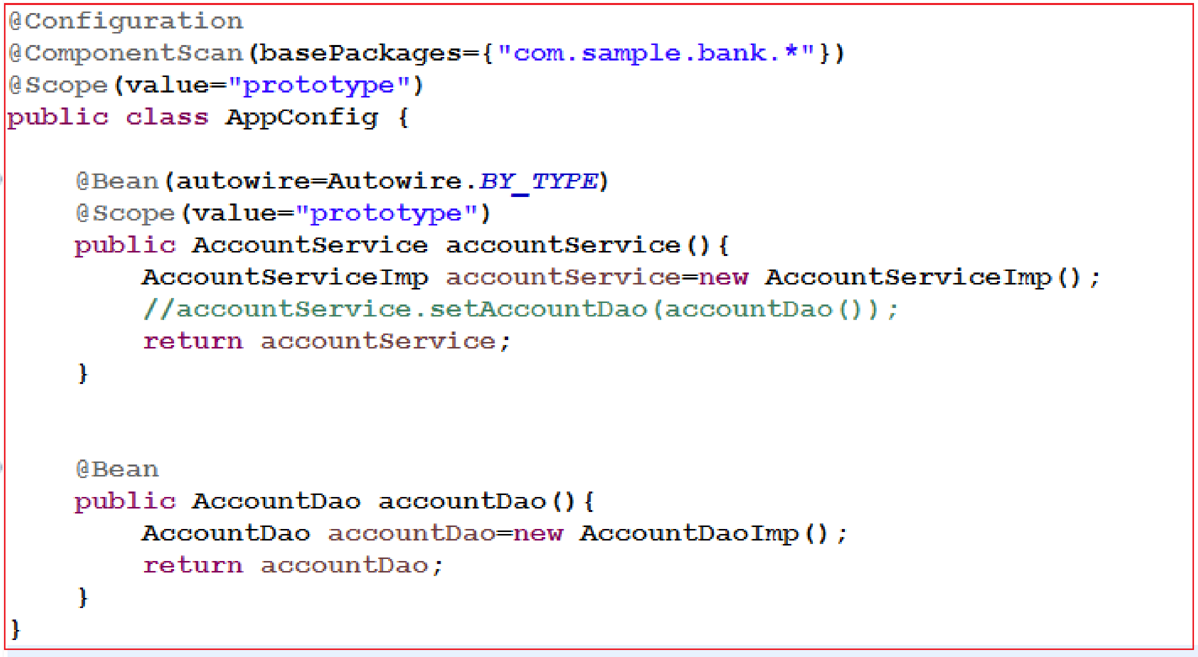
Dependency Injection xml, java configuration

XML Configuration

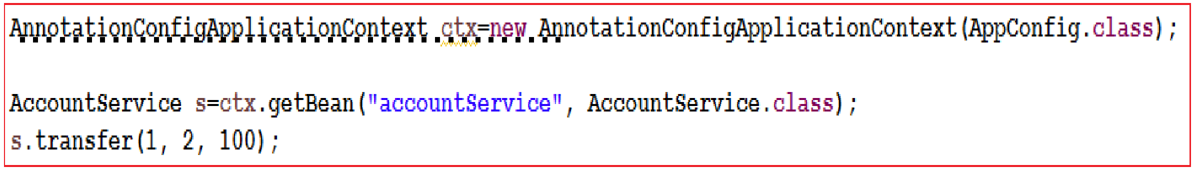


```xml
<bean id="accountService" class="com.sample.bank.model.service.AccountServiceImp" autowire="byType"/>
<bean id="accountDao1" class="com.sample.bank.model.persistance.AccountDaoImp"/>
<bean id="accountDao2" class="com.sample.bank.model.persistance.AccountDaoImp" autowire-candidate="false"/>
```

Java Configuration



```java
@Configuration
@ComponentScan(basePackages={"com.sample.bank.*"})
@Scope(value="prototype")
public class AppConfig {

    @Bean(autowire=Autowire.BY_TYPE)
    @Scope(value="prototype")
    public AccountService accountService(){
        AccountServiceImp accountService=new AccountServiceImp();
        //accountService.setAccountDao(accountDao());
        return accountService;
    }


    @Bean
    public AccountDao accountDao(){
        AccountDao accountDao=new AccountDaoImp();
        return accountDao;
    }
}
```

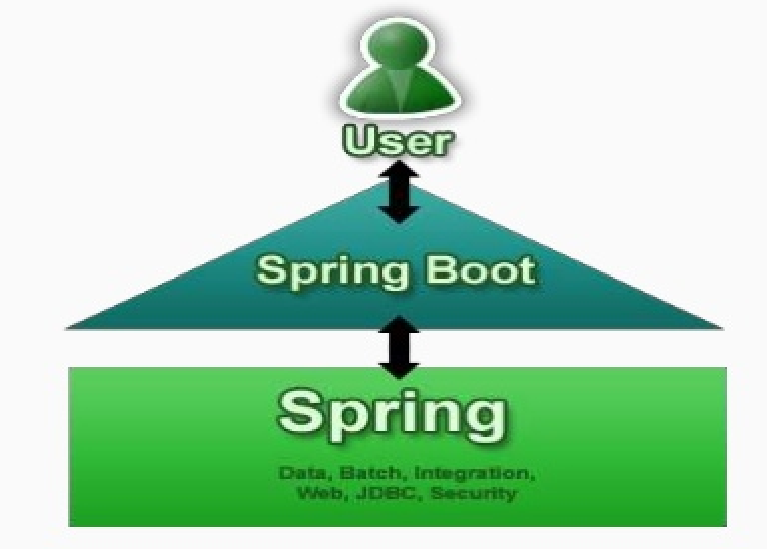


```java
AnnotationConfigApplicationContext ctx=new AnnotationConfigApplicationContext(AppConfig.class);

AccountService s=ctx.getBean("accountService", AccountService.class);
s.transfer(1, 2, 100);
```

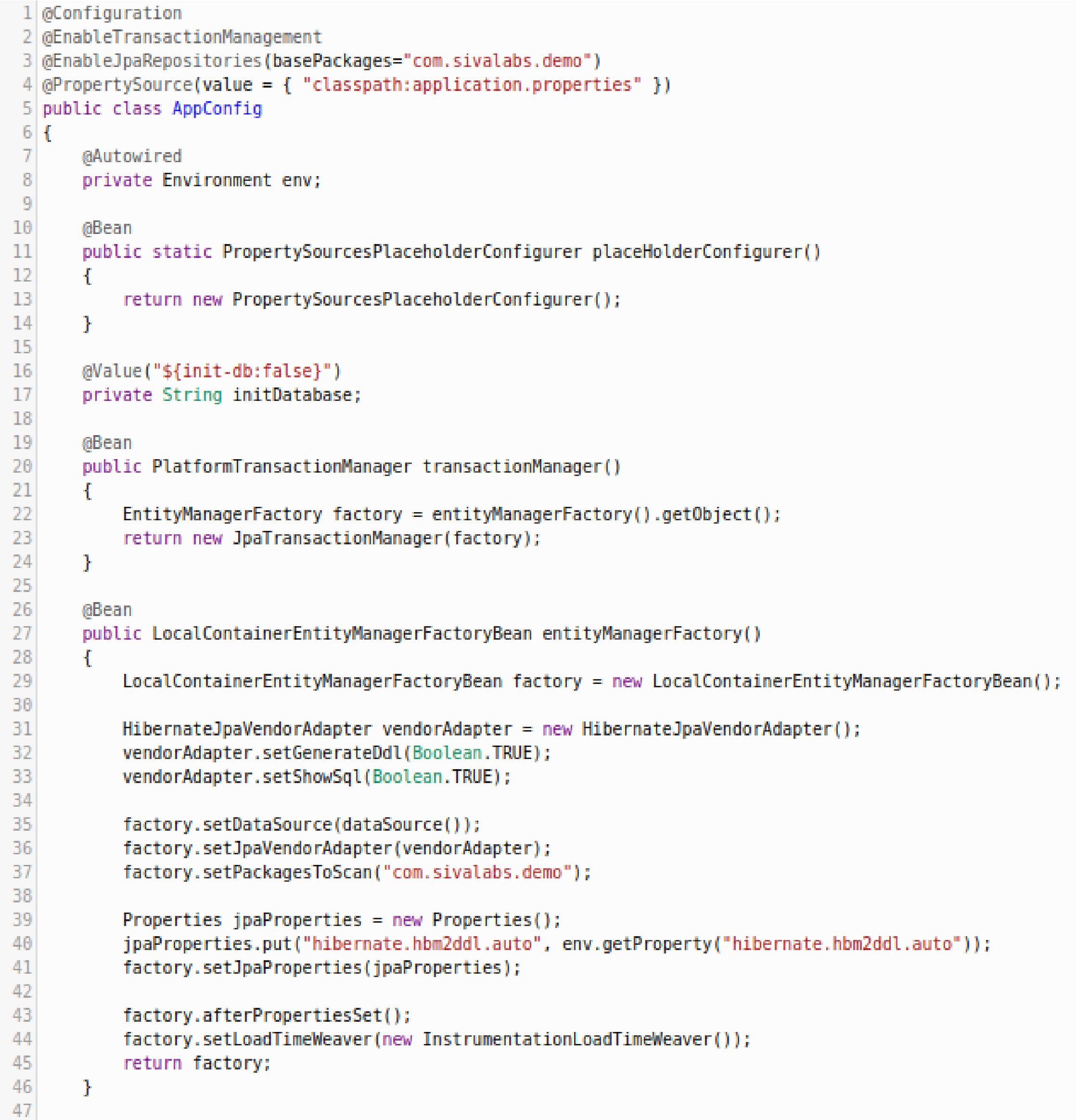
Session 1

- Introduction Spring framework, where it fits, design patterns
- Dependency Injection, Configuration-XML, Java
- **Spring Boot what why?**
- Aspect Oriented Programming, how it helps, configuration, examples
- Spring Hibernate

Spring vs Spring Boot



Why Spring Boot?



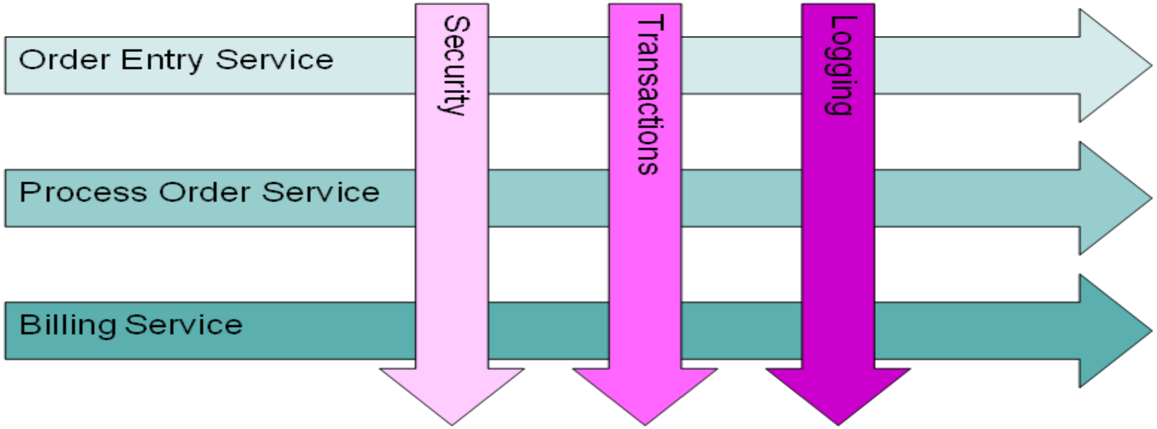
```java
@Configuration
@EnableTransactionManagement
@EnableJpaRepositories(basePackages="com.sivalabs.demo")
@PropertySource(value = { "classpath:application.properties" })
public class AppConfig
{
    @Autowired
    private Environment env;

    @Bean
    public static PropertySourcesPlaceholderConfigurer placeHolderConfigurer()
    {
        return new PropertySourcesPlaceholderConfigurer();
    }

    @Value("${init-db:false}")
    private String initDatabase;

    @Bean
    public PlatformTransactionManager transactionManager()
    {
        EntityManagerFactory factory = entityManagerFactory().getObject();
        return new JpaTransactionManager(factory);
    }

    @Bean
    public LocalContainerEntityManagerFactoryBean entityManagerFactory()
    {
        LocalContainerEntityManagerFactoryBean factory = new LocalContainerEntityManagerFactoryBean();

        HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
        vendorAdapter.setGenerateDdl(Boolean.TRUE);
        vendorAdapter.setShowSql(Boolean.TRUE);

        factory.setDataSource(dataSource());
        factory.setJpaVendorAdapter(vendorAdapter);
        factory.setPackagesToScan("com.sivalabs.demo");

        Properties jpaProperties = new Properties();
        jpaProperties.put("hibernate.hbm2ddl.auto", env.getProperty("hibernate.hbm2ddl.auto"));
        factory.setJpaProperties(jpaProperties);

        factory.afterPropertiesSet();
        factory.setLoadTimeWeaver(new InstrumentationLoadTimeWeaver());
        return factory;
    }

```

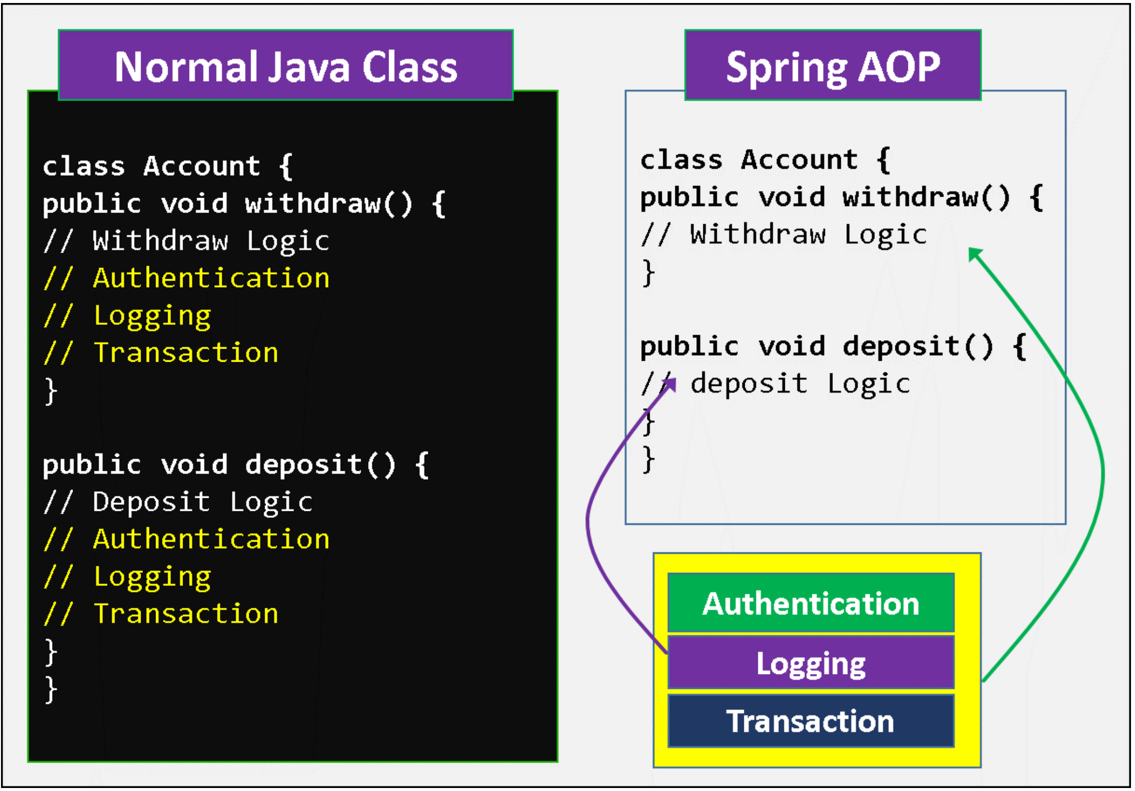
Session 1

- Introduction Spring framework, where it fits, Spring boot design patterns
- Dependency Injection, Configuration-XML, Java
- Spring Boot What Why?
- **Aspect Oriented Programming, how it helps, configuration, examples**
- Spring Hibernate

Introduction to AOP

- What is AOP?
  - AOP is a style of programming, mainly good in separating cross cutting concerns
- How AOP works?
  - Achieved usages Proxy design Pattern to separate CCC's form actual code
  - Cross Cutting Concern ?
  - Extra code mixed with the actual code is called CCC's
  - Extra code mixed with code lead to maintenance issues
    - **Logging**
    - **validations**
    - **Auditing**
    - **Security**



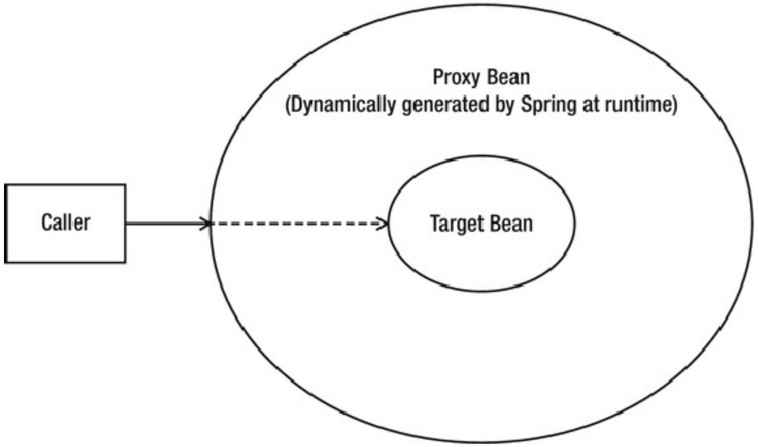


Normal Java Class

```
class Account {
public void withdraw() {
// Withdraw Logic
// Authentication
// Logging
// Transaction
}

public void deposit() {
// Deposit Logic
// Authentication
// Logging
// Transaction
}
}
```

Spring AOP

```
class Account {
public void withdraw() {
// Withdraw Logic
}

public void deposit() {
// deposit Logic
}
}
```



AOP - Definitions

- **Advice** defines what needs to be applied and when.
- **Jointpoint** is where the advice is applied.
- **Pointcut** is the combination of different joinpoints where the advice needs to be applied.
- **Aspect** is applying the Advice at the pointcuts.

Session 1

- Introduction Spring framework, where it fits, Spring boot design patterns
- Dependency Injection, Configuration-XML, Java
- Spring Boot What Why?
- Aspect Oriented Programming, how it helps, configuration, examples
- **Spring Hibernate**

Spring hibernate

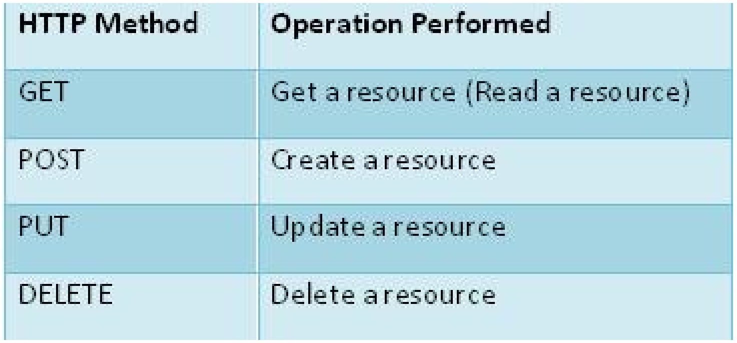
Session 2

- Spring REST
- Spring MVC
- Spring Security

Session 2

- **Spring REST**
- Spring MVC
- Spring Security

Spring REST



HTTP Method	Operation Performed
GET	Get a resource (Read a resource)
POST	Create a resource
PUT	Update a resource
DELETE	Delete a resource

Spring Annotations for REST

Annotations	Usage
@Controller	mark the class as a MVC controller
@RequestMapping	Maps the request with path
@PathVariable	Map variable from the path
@RequestBody	unmarshalls the HTTP response body into a Java object injected in the method.
@ResponseBody	marshalls return value as HTTP Response
@Configuration	Spring Config as a class

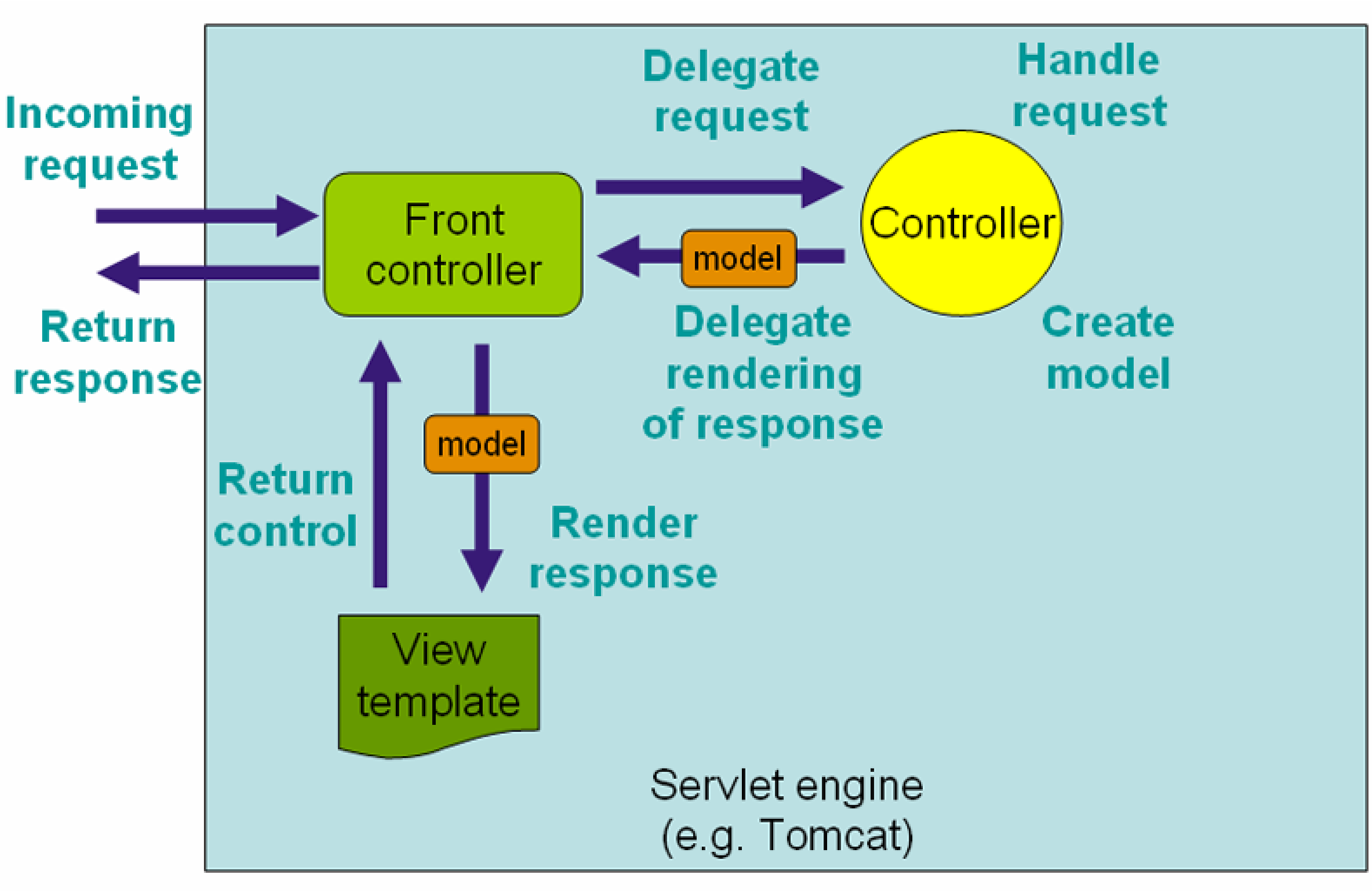
Example showing Annotations

```
@Controller
@RequestMapping(value = "/ilo")
public class iLOController
{
    @RequestMapping(value = "/server/{id}", method = RequestMethod.GET)
    public @ResponseBody Book getServer(@PathVariable String id) {
           System.out.println("------Gettting Server ------"+id);
     }
        ......
        ........
}
```

Session 2

- Spring REST
- **Spring MVC**
- Spring Security

Spring MVC

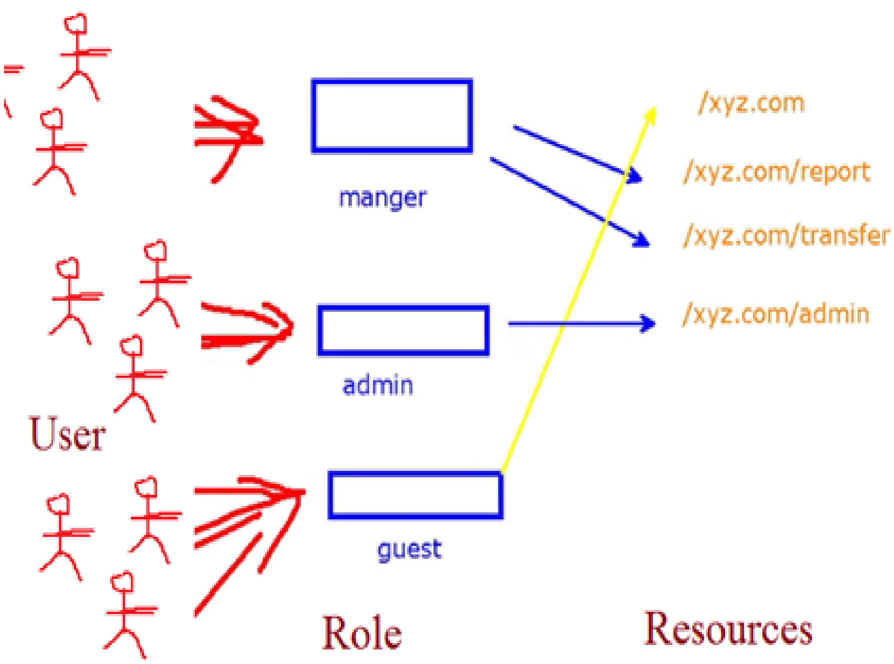

Incoming request
Return response
Front controller
Delegate request
Handle request
Controller
model
Delegate rendering of response
Create model
model
Return control
Render response
View template
Servlet engine (e.g. Tomcat)

Session 2

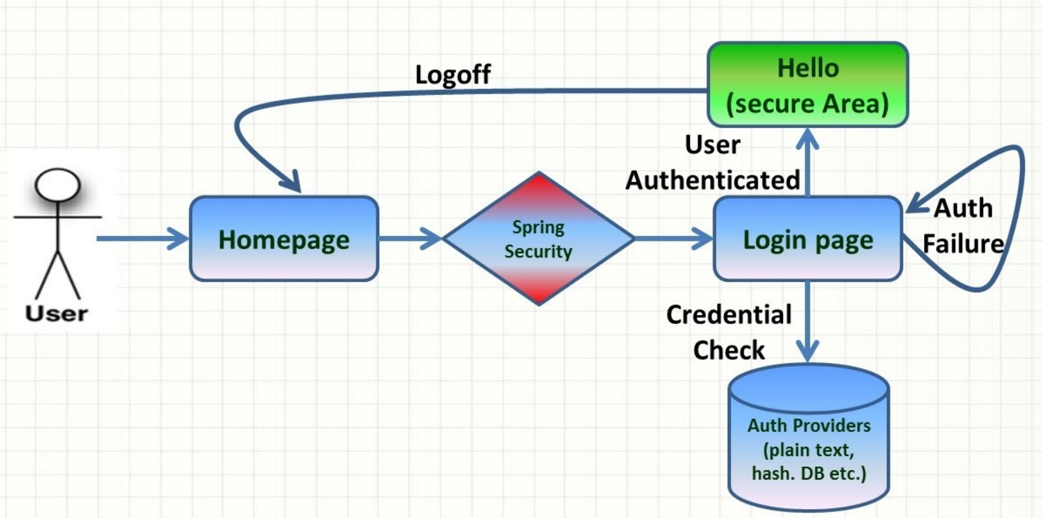
- Spring REST
- Spring MVC
- **Spring Security**

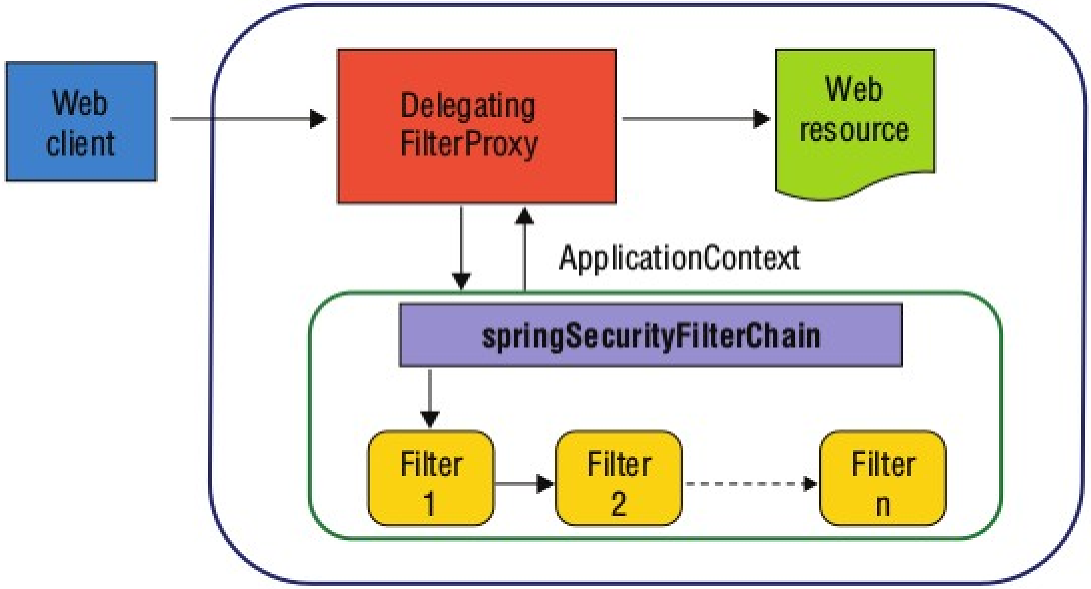
Role based Access Control RAC

- Role based Access Control (RAC)
  - As it is difficult to manage permission for each user, each user is assigned to a role and permission is set for the role
  - Authentication using Spring
    - Http Basic Authentication (uses in XML- pop up form)
    - Http form based Authentication(uses in XML- custom form)
    - Http form based Authentication(uses in DB)

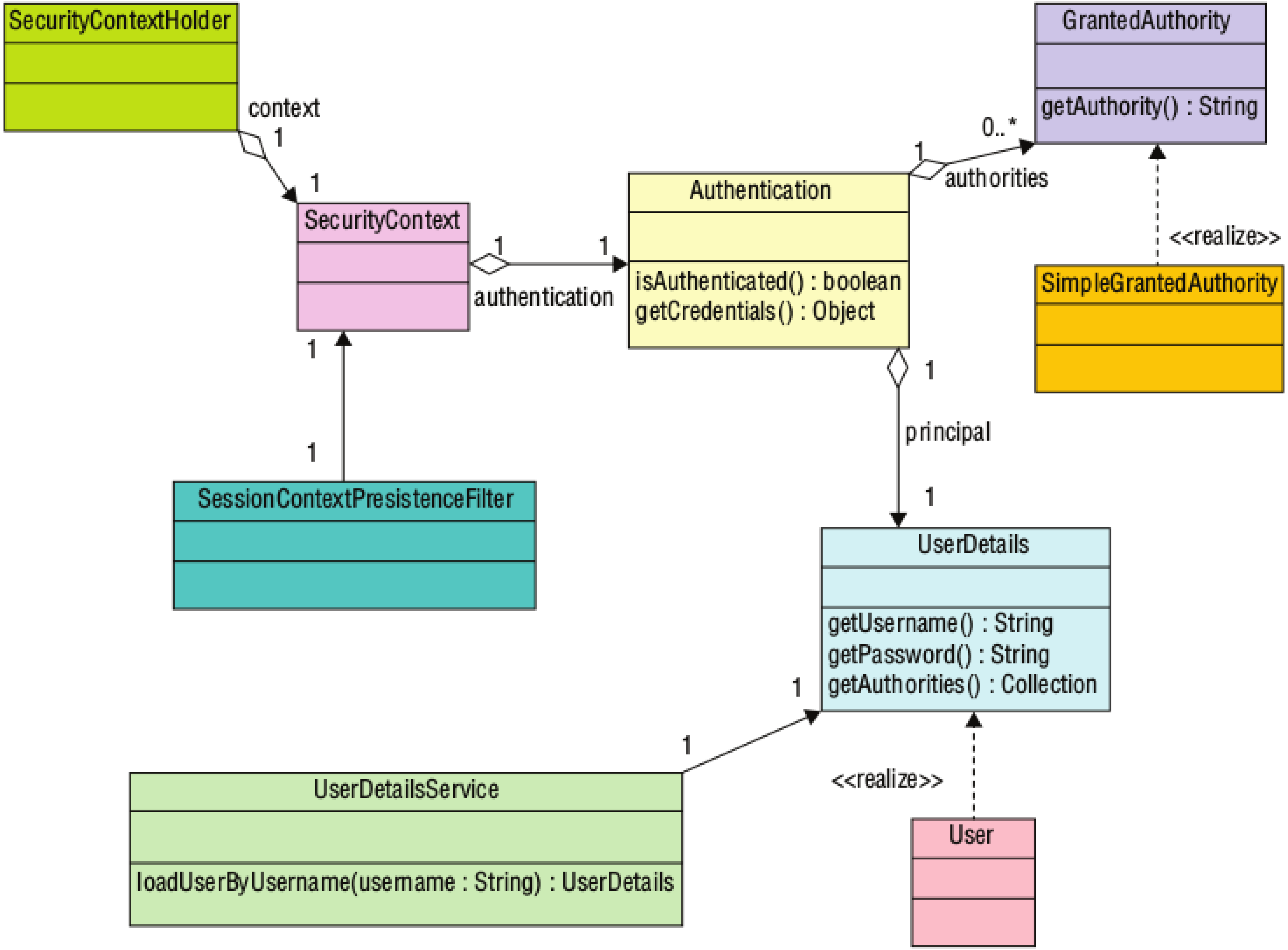


Spring security Why? How?

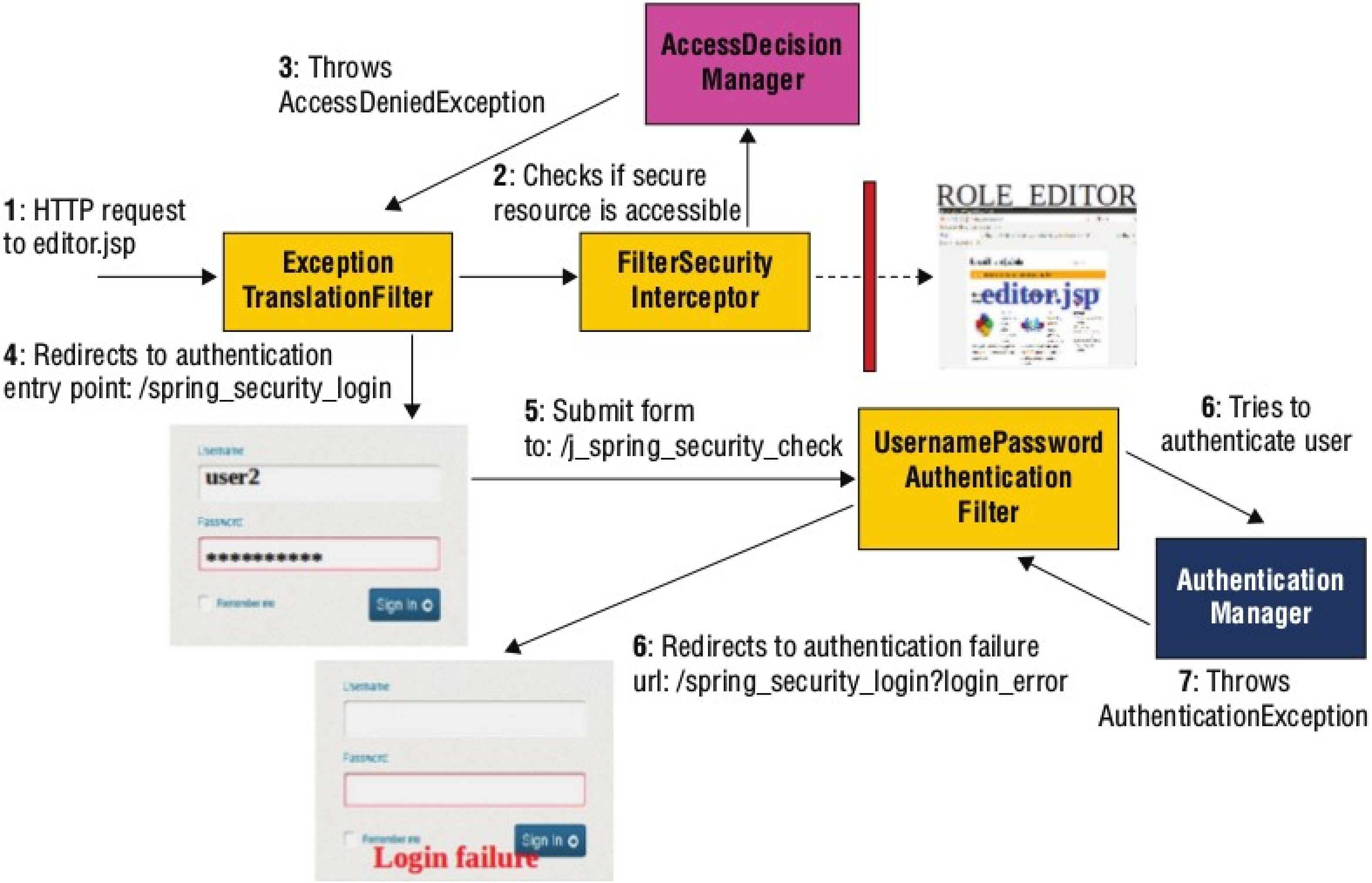




Building Blocks of Spring Security



Unsuccessful Login Flow



1: HTTP request to editor.jsp

Exception TranslationFilter

FilterSecurity Interceptor

2: Checks if secure resource is accessible

AccessDecision Manager

3: Throws AccessDeniedException

ROLE_EDITOR

editor.jsp

4: Redirects to authentication entry point: /spring_security_login

5: Submit form to: /j_spring_security_check

UsernamePassword Authentication Filter

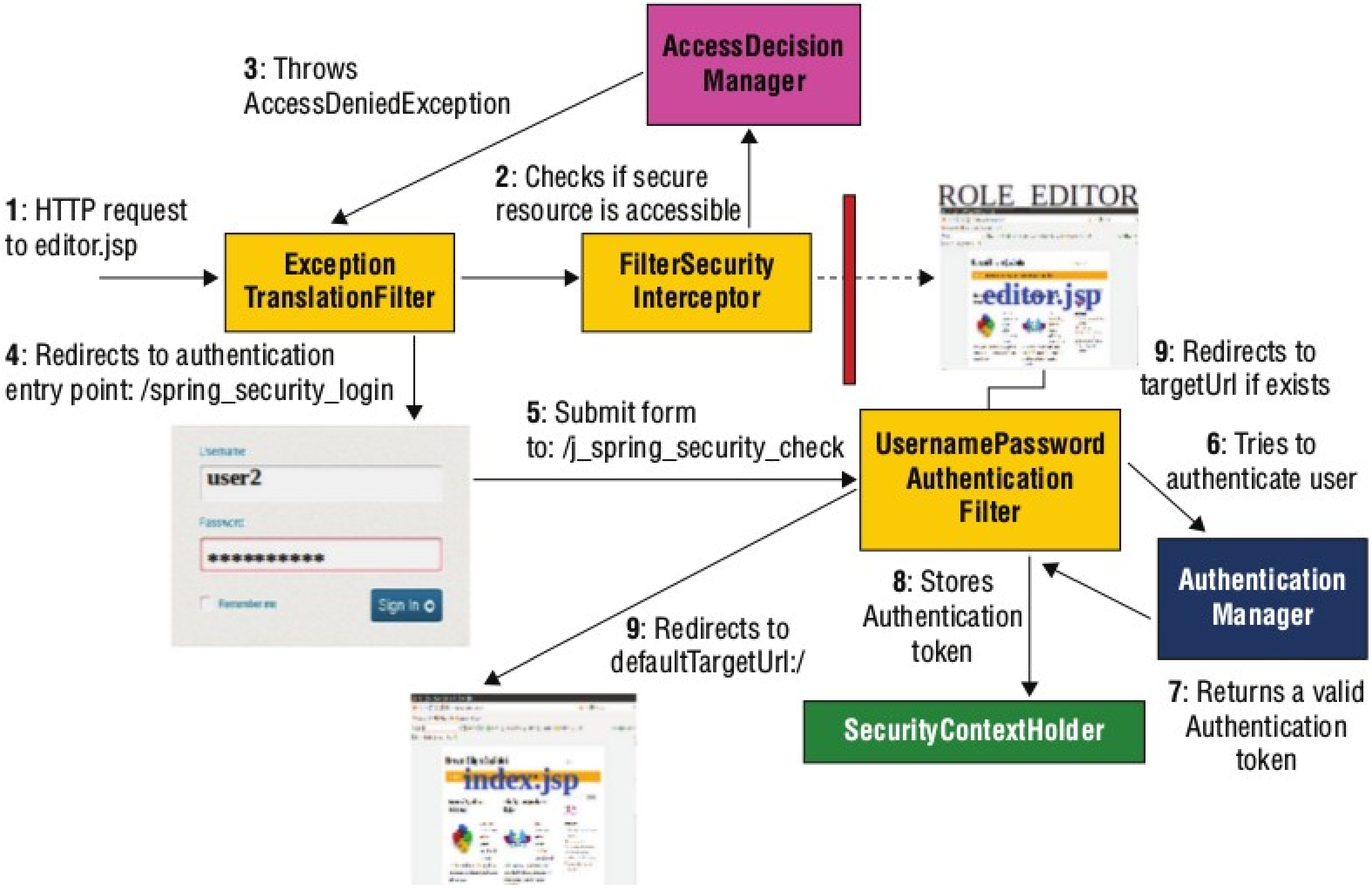
6: Tries to authenticate user

Authentication Manager

7: Throws AuthenticationException

6: Redirects to authentication failure url: /spring_security_login?login_error

Successful Login Flow



1: HTTP request to editor.jsp

2: Checks if secure resource is accessible

3: Throws AccessDeniedException

4: Redirects to authentication entry point: /spring_security_login

5: Submit form to: /j_spring_security_check

6: Tries to authenticate user

7: Returns a valid Authentication token

8: Stores Authentication token

9: Redirects to targetUrl if exists

9: Redirects to defaultTargetUrl:/

Thank You!