

Day 4: Spring Core - Policy Administration Backend

Task 1: Refactor policy-related operations to utilize Spring Beans and Dependency Injection.

```
package com.example.policy;
```

```
public class Policy {  
    private String policyNumber;  
    private String policyHolder;  
    private double premiumAmount;  
    public String getPolicyNumber() {  
        return policyNumber;  
    }  
    public void setPolicyNumber(String policyNumber) {  
        this.policyNumber = policyNumber;  
    }  
    public String getPolicyHolder() {  
        return policyHolder;  
    }  
    public void setPolicyHolder(String policyHolder) {  
        this.policyHolder = policyHolder;  
    }  
    public double getPremiumAmount() {  
        return premiumAmount;  
    }  
    public void setPremiumAmount(double premiumAmount) {  
        this.premiumAmount = premiumAmount;  
    }  
}
```

Task 2: Implement Spring validation on the server side to ensure policy data integrity.

```
package com.example.policy;
```

```

import org.springframework.validation.Errors;
import org.springframework.validation.ValidationUtils;
import org.springframework.validation.Validator;

public class PolicyValidator implements Validator {

    @Override
    public boolean supports(Class<?> clazz) {
        return Policy.class.equals(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {
        Policy policy = (Policy) target;

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "policyNumber", "policyNumber.required",
"Policy number is required.");

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "policyHolder", "policyHolder.required",
"Policy holder is required.");

        if (policy.getPremiumAmount() <= 0) {
            errors.rejectValue("premiumAmount", "premiumAmount.invalid", "Premium amount must be
greater than zero.");
        }
    }
}

```

Task 3: Set up Application Context and Bean Factory for a scalable backend structure.

```

package com.example.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import com.example.policy.PolicyService;
import com.example.policy.PolicyValidator;

@Configuration

```

```

@EnableWebMvc

@ComponentScan(basePackages = { "com.example.controller", "com.example.policy" })

public class AppConfig {

    @Bean

    public PolicyService policyService() {

        return new PolicyService(policyRepository());

    }

    @Bean

    public PolicyValidator policyValidator() {

        return new PolicyValidator();

    }

}

package com.example.config;

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
    @Override

    protected Class<?>[] getRootConfigClasses() {

        return new Class<?>[] { AppConfig.class };

    }

    @Override

    protected Class<?>[] getServletConfigClasses() {

        return null;

    }

    @Override

    protected String[] getServletMappings() {

        return new String[] { "/" };

    }

}

```