Trainer: Nilesh Ghule

Wake up from Hibernate, Spring up!!!

Agenda

- Spring exception handling
- Static resources in Web MVC
- MVC using Annotation config
- MVC security
- REST introduction
- @RestController
- RestTemplate

Spring Static resources

- Static resources like JS, CSS, images should be mapped to some location.
- <mvc:resources location="/WEB-INF/static/css/“ mapping="/css/**“/>

Spring exception handler

- Implement user defined exception class.
- In controller request handler method use @ExceptionHandler({MyException.class}).
- Configure exception mapping in spring config.

```
<bean   class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
        <property name="exceptionMappings">
                <props>
                        <prop key="java.lang.RuntimeException">error</prop>
                </props>
        </property>
        <property name="defaultErrorView" value="error" />
</bean>
```

Spring WebMVC Annotation Config

- pom.xml: properties → <failOnMissingWebXml>false</failOnMissingWebXml>
- web.xml is replaced by MyWebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer
  - Encapsulate declaration of dispatcher servlet.
  - Required for creating spring container/context.
    - getRootConfigClasses(): return config class for root webapplicationcontext.
    - getServletConfigClasses(): return config class for servlet webapplicationcontext
    - getServletMappings(): return dispatcher servlet url pattern (/)
- spring5-servlet.xml is replaced by MyWebMvcConfig implements WebMvcConfigurer
  - @Configuration
  - @ComponentScan(...): scans stereo-type annotations + other @Configuration classes.
  - @EnableWebMvc: creates AnnotationConfigWebApplicationContext.
  - @Bean: viewResolver, multipartResolver, messageSource, localeResolver, …
  - @Override: addInterceptors(): <mvc:interceptors …/> | addResourceHandlers(): <mvc:resources ... />
- MyHibernateConfig
  - @Configuration
  - @EnableTransactionManagement: <tx:annotation-driven … />
  - @Bean: dataSource, sessionFactory, transactionManager, …

Spring Application Context

- One spring application can have multiple application contexts.
    - While creating new application context, we must specify parent application context.
    - If a bean is not resolved by child application context, it will ask parent application context to resolve it.
    - Thus child context can access beans of parent; however reverse is not true.

Spring Web Application Context

- Each spring web application needs at least one WebApplicationContext.
- In simple spring MVC web application, the dispatcher servlet is responsible for creating spring webapplicationcontext.
  - This context is responsible for MVC as well as dependency injection.
- If spring is to be used for DI along with any existing MVC framework like struts or JSF, we cannot use dispatcher servlet.
  - Use ServletContextListener (named as "ContextLoaderListener") is registered in web.xml, which is responsible for creating spring webapplicationcontext.
  - This webapplicationcontext will handle DI and struts/JSF controller will handler MVC.
- In typical Spring MVC applications we can have ContextLoaderListener to create "root" webapplicationcontext and one/more dispatcher servlets for managing MVC.
- The dispatcher servlets also create their own webapplicationcontext, which are child(s) of "root" webapplicationcontext.

Spring WebMVC Security

- Authentication
  - Checking user/client credentials.
  - Who am I?
- Authorization:
  - Checking whether user/client is allowed to access the resource.
  - Am I authorized?
- Role Based Security:
  - Each user is assigned ROLE. A user may be in multiple ROLEs.
  - Each ROLE is allowed access for certain resources & denied access for certain resources.

Spring WebMVC Security

REST services

- REpresentation State Transfer
- Protocol to invoke web services from any client.
  - Client can use any platform/language.
- REST works on top of HTTP protocol.
  - Can be accessed from any device which has internet connection.
  - REST is lightweight (than SOAP) – XML or JSON.
  - Uses HTTP protocol request methods
    - GET: to get records
    - POST: to create new record
    - PUT: to update existing record
    - DELETE: to delete record

Spring REST services

- Based on top of Spring Web MVC.
- Maven dependency: jackson-databind and jackson-databind-xml (if need XML)
- HttpMessageConverter beans:
  - MappingJackson2HttpMessageConverter, MappingJackson2XmlHttpMessageConverter
  - @Override configureMessageConverters()
- Using @Controller
  - @GetMapping, @PostMapping, @PutMapping, @DeleteMapping or @RequestMapping
  - @RequestBody, @ResponseBody / ResponseEntity<>
  - @PathVariable, @RequestParam
- Using @RestController
  - @GetMapping, @PostMapping, @PutMapping, @DeleteMapping or @RequestMapping
  - @RequestBody, ResponseEntity<>
  - @PathVariable, @RequestParam
- To manipulate JSON response use @JsonProperty or @JsonIgnore.

Spring REST services

- Spring REST services can be invoked by RestTemplate.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>