The QuoteServices App Case Study: Base* Super Class Structure & Functionality

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

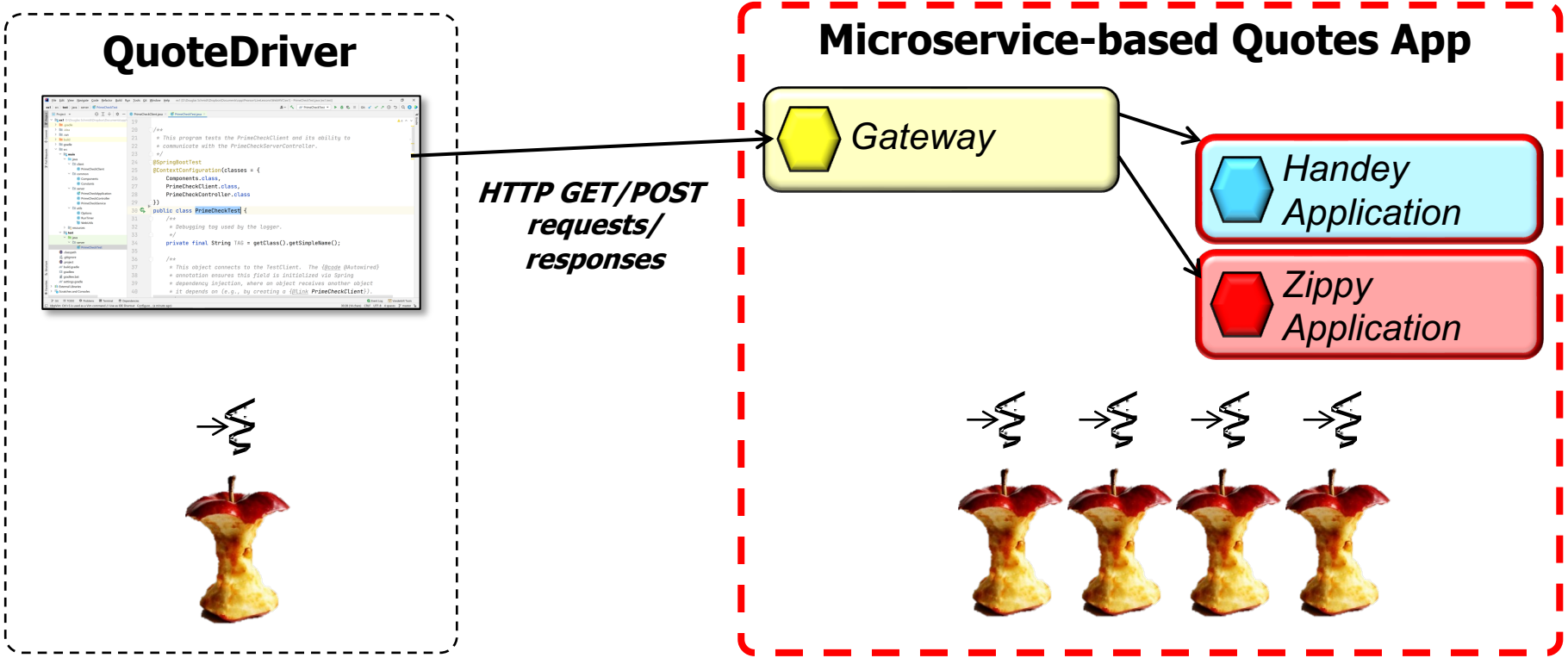
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of the super classes BaseApplication, BaseController, & BaseService



These super classes provide customizable foundation for the Quote microservices

Structure & Functionality of the BaseApplication

Structure & Functionality of the BaseApplication

- BaseApplication provides the basis for a generic Spring microservice

```
public class BaseApplication {


  public static void run(Class<?> clazz, String[] args) {
    var name = getName(clazz);
    var app = new SpringApplicationBuilder(clazz)
      .properties(singletonMap("spring.application.name",
                               name))
      .build();
    app.setAdditionalProfiles(name);
    app.setLazyInitialization(true);
    app.run(args);
  }
  ...
```

This class is extended by Handey Application & ZippyApplication

See microservices/src/main/java/edu/vandy/quoteservices/common/BaseApplication.java

Structure & Functionality of the BaseApplication

- BaseApplication provides the basis for a generic Spring microservice

```
public class BaseApplication {


  public static void run(Class<?> clazz, String[] args) {
    var name = getName(clazz);
    var app = new SpringApplicationBuilder(clazz)
      .properties(singletonMap("spring.application.name",
                               name))
      .build();
    app.setAdditionalProfiles(name);
    app.setLazyInitialization(true);
    app.run(args);
  }
  ...
```

Builds a Spring Boot micro service with the clazz name

Structure & Functionality of the BaseApplication

- BaseApplication provides the basis for a generic Spring microservice

```
public class BaseApplication {


  public static void run(Class<?> clazz, String[] args) {
    var name = getName(clazz);
    var app = new SpringApplicationBuilder(clazz)
      .properties(singletonMap("spring.application.name",
                               name))
    ...
  }
  private static String getName(Class<?> clazz) {
    String pkg = clazz.getPackage().getName();
    return pkg.substring(pkg.lastIndexOf('.') + 1);
  }
```

Get the name of the microservice application

Structure & Functionality of the BaseController

Structure & Functionality of the BaseController

- BaseController maps HTTP GET/POST requests to endpoint handler methods

```
public abstract class BaseController<T> {
   ...
   @Autowired
   BaseService<T> mService;

   @GetMapping(GET_ALL_QUOTES)
   public T getAllQuotes)
   { return mService.getAllQuote(); }

   ...
   @PostMapping(POST_SEARCHES)
   public T search(@RequestBody List<String> queries,
                   Boolean parallel)
   { return mService.search(queries, parallel); }
}
```

This class is extended by HandeyController & ZippyController

See microservices/src/main/java/edu/vandy/quoteservices/common/BaseController.java

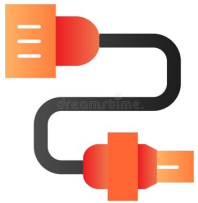
Structure & Functionality of the BaseController

- BaseController maps HTTP GET/POST requests to endpoint handler methods

```
public abstract class BaseController<T> {
   ...
   @Autowired
   BaseService<T> mService;

   @GetMapping(GET_ALL_QUOTES)
   public T getAllQuotes)
   { return mService.getAllQuote(); }

   ...
   @PostMapping(POST_SEARCHES)
   public T search(@RequestBody List<String> queries,
                   Boolean parallel)
   { return mService.search(queries, parallel); }
}
```



This field is auto-wired by Spring's dependency injection framework

See www.baeldung.com/spring-autowire

Structure & Functionality of the BaseController

- BaseController maps HTTP GET/POST requests to endpoint handler methods

```
public abstract class BaseController<T> {
   ...
   @Autowired
   BaseService<T> mService;

   @GetMapping(GET_ALL_QUOTES)
   public T getAllQuotes)
   { return mService.getAllQuote(); }

   ...
   @PostMapping(POST_SEARCHES)
   public T search(@RequestBody List<String> queries,
                   Boolean parallel)
   { return mService.search(queries, parallel); }
}
```

These methods forward to the Handey Service methods & return the results back to the client

Structure & Functionality of the BaseController

- BaseController maps HTTP GET/POST requests to endpoint handler methods

```
public abstract class BaseController<T> {
   ...
   @Autowired
   BaseService<T> mService;
   ...
   @GetMapping(GET_ALL_QUOTES)
   public T getAllQuotes)
   { return mService.getAllQuote(); }

   ...
   @PostMapping(POST_SEARCHES)
   public T search(@RequestBody List<String> queries,
                   Boolean parallel)
   { return mService.search(queries, parallel); }
}
```

These annotations map HTTP GET & POST requests onto endpoint handler methods

See www.baeldung.com/spring-new-requestmapping-shortcuts

Structure & Functionality of the BaseController

- BaseController maps HTTP GET/POST requests to endpoint handler methods

```
public abstract class BaseController<T> {
   ...
   @Autowired
   BaseService<T> mService;
   ...
   @GetMapping(GET_ALL_QUOTES)
   public T getAllQuotes)
   { return mService.getAllQuote(); }

   ...
   @PostMapping(POST_SEARCHES)
   public T search(@RequestBody List<String> queries,
                   Boolean parallel)
   { return mService.search(queries, parallel); }
}
```

These strings automatically identify & route to endpoint handler methods from incoming GET & POST requests

See www.baeldung.com/spring-new-requestmapping-shortcuts

Structure & Functionality of the BaseController

- BaseController maps HTTP GET/POST requests to endpoint handler methods

```
public abstract class BaseController<T> {
   ...
   @Autowired
   BaseService<T> mService;
   ...
   @GetMapping(GET_ALL_QUOTES)
   public T getAllQuotes)
   { return mService.getAllQuote(); }

   ...
   @PostMapping(POST_SEARCHES)
   public T search(@RequestBody List<String> queries,
                   Boolean parallel)
   { return mService.search(queries, parallel); }
}
```

This annotation automatically deserializes the inbound HttpRequest body onto a Java object

See www.baeldung.com/spring-request-response-body

Structure & Functionality of the BaseController

- BaseController maps HTTP GET/POST requests to endpoint handler methods

```
public abstract class BaseController<T> {
   ...
   @Autowired
   BaseService<T> mService;
   ...
   @GetMapping(GET_ALL_QUOTES)
   public T getAllQuotes)
   { return mService.getAllQuote(); }

   ...
   @PostMapping(POST_SEARCHES)
   public T search(@RequestBody List<String> queries,
                   Boolean parallel)
   { return mService.search(queries, parallel); }
}
```

Note the generic param that's used to define the return type

Generics enables BaseController to work for both classic & reactive Java types

Structure & Functionality of the BaseService

Structure & Functionality of the HandeyService

- BaseService defines a reusable/customizable Quote service API

```
public interface BaseService<T> {

  T getAllQuotes();

  T postQuotes(List<Integer> quoteIds,
               Boolean parallel);

  T search(List<String> queries,
           Boolean parallel);
  ...
}
```

This interface is extended by HandeyService & ZippyService

See microservices/src/main/java/edu/vandy/quoteservices/common/BaseService.java

Structure & Functionality of the HandeyService

- BaseService defines a reusable/customizable Quote service API

```
public interface BaseService<T> {

  T getAllQuotes();

  T postQuotes(List<Integer> quoteIds,
               Boolean parallel);

  T search(List<String> queries,
           Boolean parallel);
  ...
}
```

These methods are overridden & implemented in the subclasses

Structure & Functionality of the HandeyService

- BaseService defines a reusable/customizable Quote service API

```
public interface BaseService<T> {

  T getAllQuotes();

  T postQuotes(List<Integer> quoteIds,
               Boolean parallel);

  T search(List<String> queries,
           Boolean parallel);
  ...
}
```

This generic param is used to define the return type

Generics enables BaseService to work for both classic & reactive Java types

End of the QuoteServices App Case Study: Base* Super Class Structure & Functionality