Hewlett Packard Enterprise

NonStop Technical Boot Camp 2023 TBC23-TB60 Build a REpresentational State Transfer (REST) API microservice on NonStop in 15 minutes

John Zimsky, NonStop Advanced Technology Center

September 2023

Forward-looking statements

This is a rolling (up to three year) Roadmap and is subject to change without notice

This document contains forward looking statements regarding future operations, product development, product capabilities and availability dates. This information is subject to substantial uncertainties and is subject to change at any time without prior notification. Statements contained in this document concerning these matters only reflect Hewlett Packard Enterprise's predictions and / or expectations as of the date of this document and actual results and future plans of Hewlett Packard Enterprise may differ significantly as a result of, among other things, changes in product strategy resulting from technological, internal corporate, market and other changes. This is not a commitment to deliver any material, code or functionality and should not be relied upon in making purchasing decisions.

Agenda

Why use a framework?

Getting started

Building a “hello world” service

Beyond “hello world”

Where else can you deploy on the NonStop

Thanks for attending

NonStop Partnership– It's a Beautiful Thing!

4techsoftware · abat · absyss · ACI UNIVERSAL PAYMENTS · ADVANCED SYSTEMS CONCEPTS, INC. · amdocs · Ascert · asseco · axway · BHMI · BlackWood Systems, Inc. · bmc

Bowden Systems Inc. · BPC · BrightStrand INTERNATIONAL · ca technologies A Broadcom Company · CAIL · Canam Software XML and reporting solutions! · Capgemini · CAPSYS · CARR❖SCOTT SOFTWARE INCORPORATED · Thycotic Centrify · CGI · CIRCA · DATA443

comforte · concerto Payment Systems Architects · connect Your Independent Hewlett Packard Enterprise Technology User Community · continuous NonStop IT Solutions · Cornerstone SOFTWARE · crossflo systems · CRYSTAL POINT® · CS software · CSP · Daiwa Institute of Research

DXC.technology · ePROTEA FINEXUS · esq · ETI-NET · EUROMETRIC · eztips · FIS · FSS A Payment Systems Company · GEMINI Innovation & Leadership · GRAVIC® HPE Shadowbase · greenHouse

IBM · IDELJI · IDMWORKS · IKAN · INETCO Every transaction tells a story® · infra soft · intel · INTELLIGENT WAVE INC. · ir · ITP SW Systeme · Japan Research Institute · Jit2000

KAUFFMANN ENTERPRISE · KCS (주)케이씨에스 · Knightcraft Technology Information Security Specialists · KSS 京信システムサービス Kyoshin System Service Co., Ltd. · lusis Payments · m2systems Transacting data, transforming companies. · MICRO FOCUS · MVI TECHNOLOGIES · NCI Network Concepts, Inc. · netweave INTEGRATED SOLUTIONS

Nexbridge · NonStopInsider · nti · NuWave MISSION-CRITICAL ENTERPRISE INTEGRATION · ODYSSEY INFORMATION SERVICES · OmniPayments · ORACLE · OSS Nokalva® · PAYXUS · PAYX · Pragnya solutions

PrimeCode Services Inc. · Progress® formerly ipswitch® · PYALLA TECHNOLOGIES · Qlik · QSA · River Rock Software · Rocket. · Scipar, inc. · SMX I.T. SOLUTIONS PROVIDER Services & Consulting, Inc. · SI SOFTWARE INTEGRATORS

striim · Symmetryx · 凌群電腦 THE SYSCOM GROUP · TANDsoft NonStop Interception & Availability Solutions · tcm · THALES · TIC TICSOFTWARE NonStop Innovation · TIS TIS INTEC Group · BAN BOTTLENECKS® Transaction Design, Inc. · TriCom · TrueByte

TRUSTED SECURITY SOLUTIONS · U.S. Consulting Services, Inc. www.uscsi.com NonStop Solutions · utimaco® · V2verify™ · Verifone® · WORLDLINE · XID TECHNOLOGY SOLUTIONS · XYPRO Mission Critical Security

Spring Boot on NonStop
Why use a framework?

- Steps involved in providing a service written from scratch

1. Create a socket and post a listen
2. Read inbound request
3. Route request to the appropriate function
4. Parse the data in the request
5. Do work
6. Build a reply
7. Send reply to the requestor

- And make this all run as a threaded application to handle concurrent requests

Spring Boot on NonStop
Why use a framework?

- In general, many of the “How do I do XYZ?” questions can be answered with “there is a framework for that!”
- From the Spring Boot site (https://spring.io/projects/spring-boot)

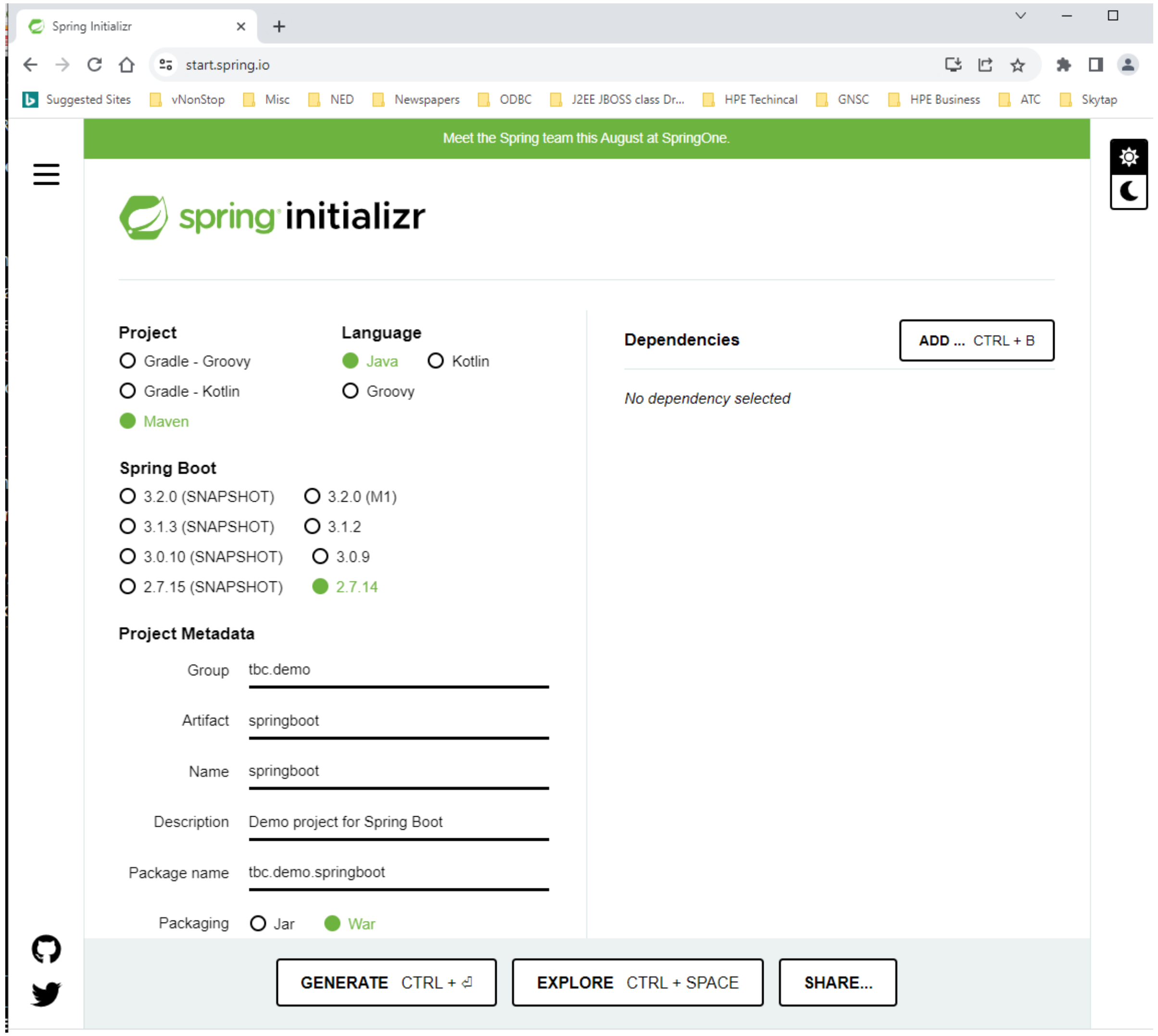
Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

- Spring Boot allows you to quickly and easily deploy a microservice without having to code everything by hand
- It only requires java
- As an open-source project, it is supported by the open-source community.
- HPE does support NSJ and will address any issues in that product.

Spring Boot on NonStop
Getting started

- You can either use your favorite Integrated Development Environment (IDE) or use the “spring initilizr” site https://start.spring.io
- For this demo we will use

1. Project - Maven
2. Language - Java
3. Spring Boot - 2.7.14
4. Packaging - War



Project: Gradle - Groovy, Gradle - Kotlin, [✓ Maven]

Language: [✓ Java], Kotlin, Groovy

Spring Boot: 3.2.0 (SNAPSHOT), 3.2.0 (M1), 3.1.3 (SNAPSHOT), 3.1.2, 3.0.10 (SNAPSHOT), 3.0.9, 2.7.15 (SNAPSHOT), [✓ 2.7.14]

Project Metadata

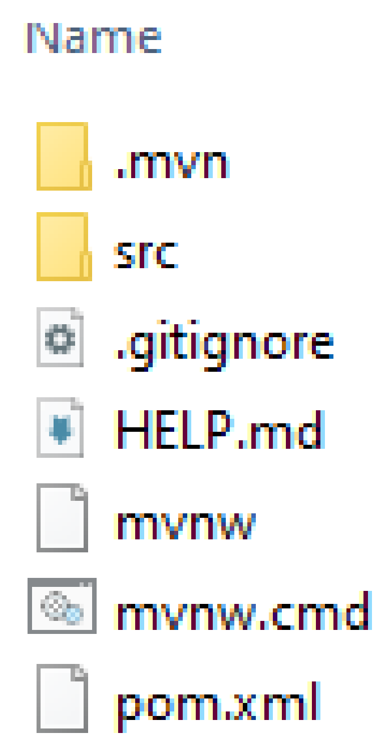
Group	tbc.demo
Artifact	springboot
Name	springboot
Description	Demo project for Spring Boot
Package name	tbc.demo.springboot
Packaging	Jar [✓ War]

Dependencies — ADD ... CTRL + B

No dependency selected

Spring Boot on NonStop
Getting started

- The initializer will download a ZIP file that will contain a ready to build application

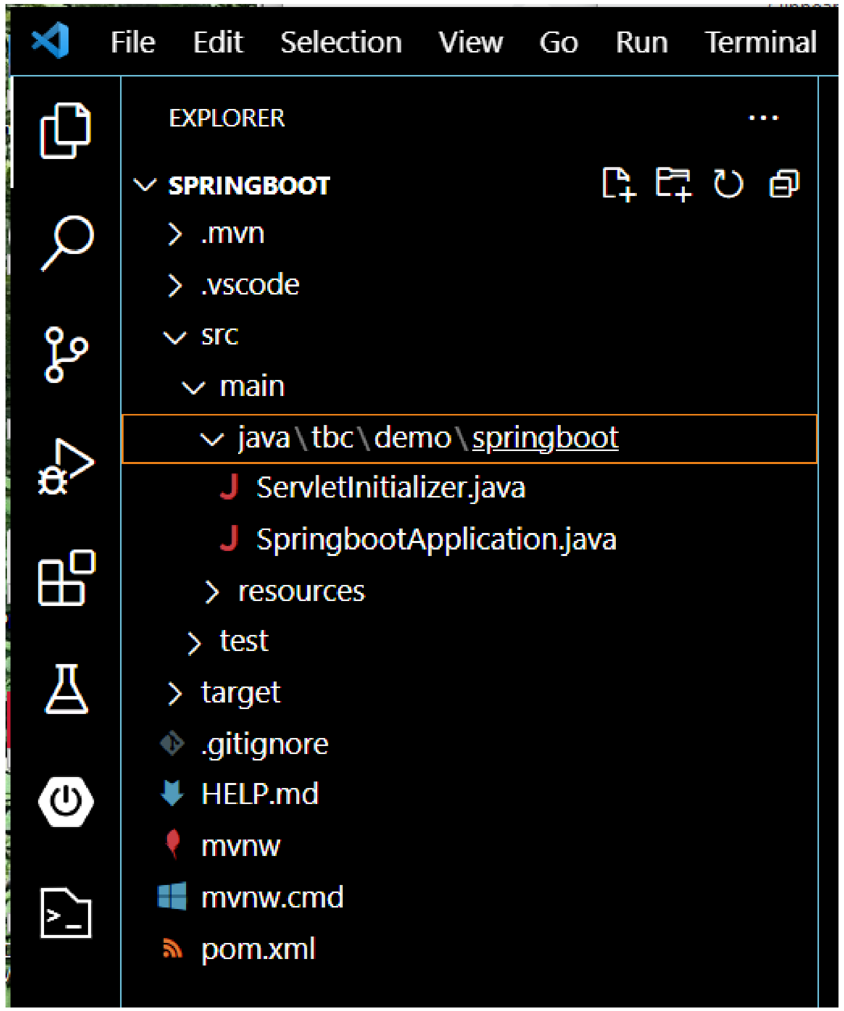


Name

.mvn
src
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml

Spring Boot on NonStop
Building a “hello world” service

- You can do your builds in OSS as maven is also pure java code and will run on the NonStop platform.
- Using an IDE does have some real benefits over just a standard text editor.
- My IDE of choice for the moment is VSCode


File Edit Selection View Go Run Terminal

EXPLORER

SPRINGBOOT
.mvn
.vscode
src
main
java \ tbc \ demo \ springboot
ServletInitializer.java
SpringbootApplication.java
resources
test
target
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml

Spring Boot on NonStop
Building a “hello world” service

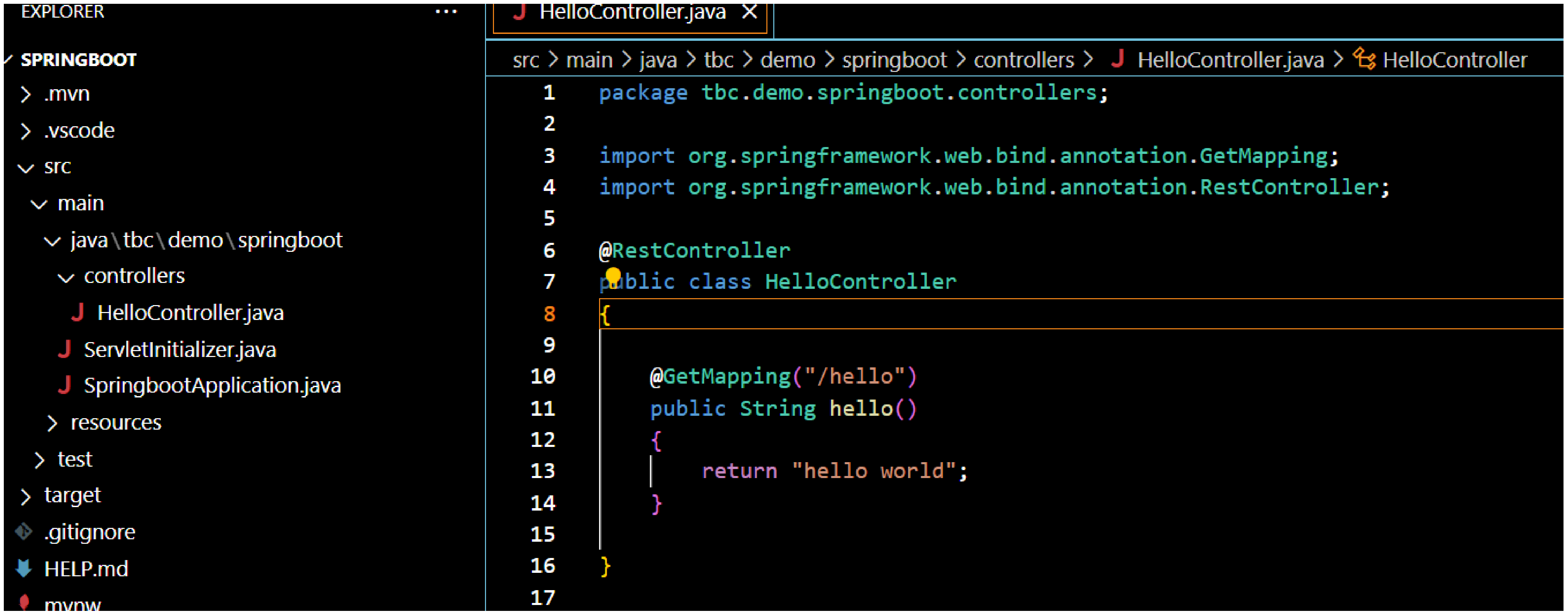
- Spring Boot has the concept of Representational State Transfer (REST) Controllers where you build the logic to handle requests, for “hello world” that is as simple as:

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController
{

   @GetMapping("/hello")
   public String hello()
   {
      return "hello world";
  }
```

Spring Boot on NonStop

Building a “hello world” service

- And in VSCode



```java
package tbc.demo.springboot.controllers;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController
{

    @GetMapping("/hello")
    public String hello()
    {
        return "hello world";
    }

}

```

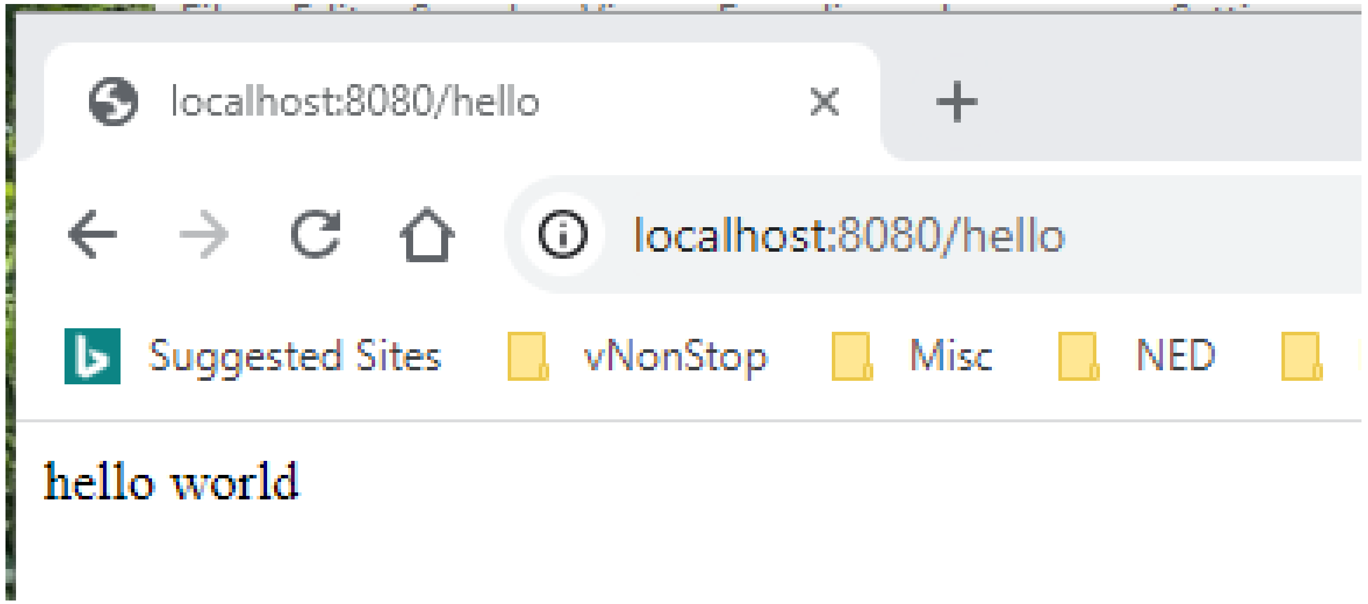
Spring Boot on NonStop

Building a “hello world” service

- And with the maven command:

```
mvn spring-boot:run
```

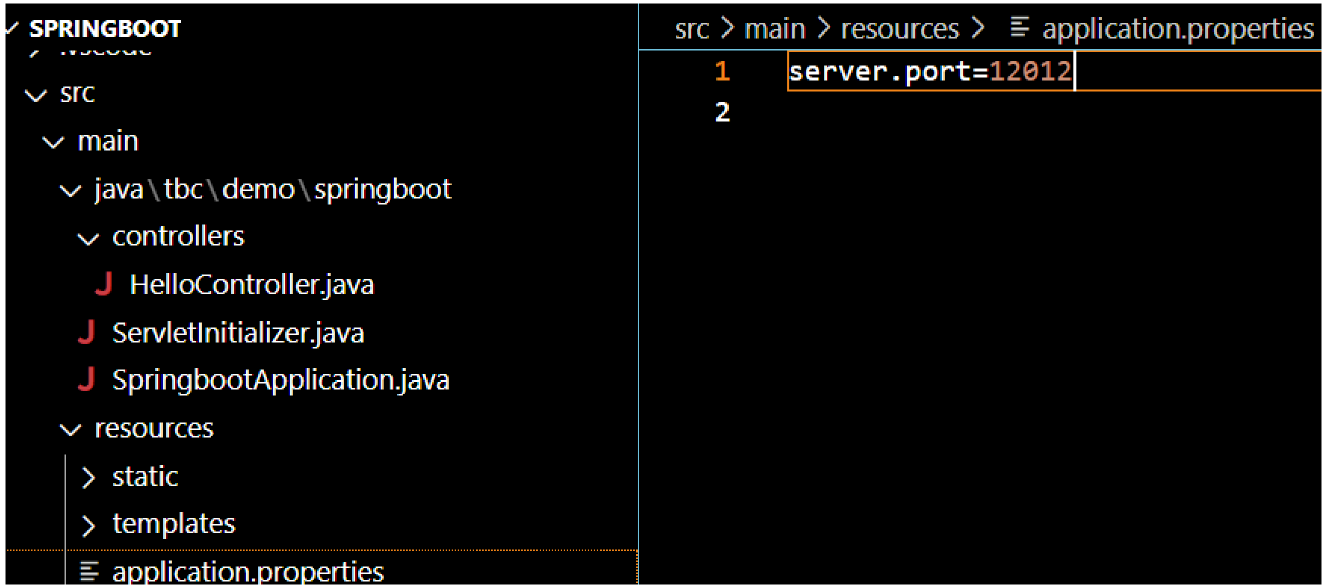
- You can start the application on your workstation and access hello world



hello world

Spring Boot on NonStop
Beyond “hello world”

- By default, the application uses port 8080 like many over frameworks.
- That can be easily changed by setting server.port in the applications properties file.


```
SPRINGBOOT
  .vscode
  src
    main
      java\tbc\demo\springboot
        controllers
          HelloController.java
        ServletInitializer.java
        SpringbootApplication.java
      resources
        static
        templates
        application.properties
```

src > main > resources > application.properties

```
server.port=12012
```

Spring Boot on NonStop
Beyond “hello world”

- To do “real work” with this application we need to define two additional java objects:

1. We need to define an entities that describes the format of the inbound and outbound JSON payloads. For our example we will do this with one entity
2. We need to define the controller that will do the work

- Neither of these objects require a lot of user code

Spring Boot on NonStop
Beyond “hello world”

```
public class Employee
{

   private Integer empid;
   private String firstname;
   private String lastname;
   private int departid;

   public Employee()
   {
                  // default constructor
   }

   public Employee(int empid, String firstname, String lastname, int departid)
   {
      this.setEmpid(empid);
      this.setFirstname(firstname);
      this.setLastname(lastname);
      this.setDepartid(departid);
   }

// getters and setters
```

Spring Boot on NonStop

Beyond “hello world”

```
@RestController
public class workController
{

    @PostMapping(path = "/doWork", consumes = MediaType.APPLICATION_JSON_VALUE)
    public Employee doWork(@RequestBody Employee employee)
    {
        Employee replyEmployee = new Employee();

        int empid = employee.getEmpid();

        replyEmployee.setEmpid(empid);
        replyEmployee.setFirstname("John");
        replyEmployee.setLastname(System.getProperty("os.name"));
        replyEmployee.setDepartid(1);

        return replyEmployee;

    }
```

Spring Boot on NonStop
Beyond “hello world”

- The input JSON payload:

```
{
   "empid" : 1
}
```

- The curl command

```
curl -X POST -H "Content-Type: application/json" http://localhost:12012/doWork -d @doWork.json
```

- The output

```
{"empid":1,"firstname":"John","lastname":“Windows 10","departid":1}
```

Spring Boot on NonStop
Beyond “hello world”

- The WAR file springboot-0.0.1-SNAPSHOT.war in the target directory can be pushed to any platform and run from a command line. For example, on my lab system:

```
java -jar springboot-0.0.1-SNAPSHOT.war
```

- Curl command changes to

```
curl -X POST -H "Content-Type: application/json" http://TBC1:12012/doWork -d @doWork.json
```

- Output is the similar, note the lastname now points to NonStop:

```
{"empid":1,"firstname":"John","lastname":“"NONSTOP_KERNEL","departid":1}
```

Spring Boot on NonStop

How else can you deploy on the NonStop

- You can put WAR file in a TS/MP configuration as a server class using TCPIP^FILTER^KEY to all multiple servers to listen on the same port:

```
set server define =PTCPIP^FILTER^KEY,CLASS MAP,FILE \$OSS.JZSPRING.KEY
```

- As this will be a pure socket application without any TS/MP API interface (SERVERCLASS_SEND_) TS/MP will not be able to the starting/stopping of dynamic servers in the server class. You can use NonStop Middleware Elasticity Framework (NSMEF) to provide for dynamic server control. That would be a topic for another presentation.

Spring Boot on NonStop

Where else can you deploy on the NonStop

- You can put the WAR file in a Pathway using NSJ Infrastructure (NSJI) and access them directly from NonStop HTTP Server (NSHTTP). This gives you the all the security of NSHTTP and since NSJI is in use you get Pathway to dynamically start servers when needed.

- The settings need to invoke JI are this env:

JI_ENABLE=true

- And this –D flag

-Dji.mapping.file=./ji.prop

Spring Boot on NonStop

Where else can you deploy on the NonStop

- This ji.prop file contains:

```
server_socket-0.0.0.0:12012 pathsend_qualifier=**ANY-DIALOG**:open_qualifier=**ANY-QUALIFIER**:mode=REQUEST_RESPONSE:single_dialog=true
```

Which tells the JI framework that all listens posted on port 12012 are to be converted to READUPDATEX() requests on $RECEIVE

Spring Boot on NonStop
Where else can you deploy on the NonStop

- Then in the NSHTTP/<deployment>/conf/proxy directory you would add a conf file, in our setup we named it sb.conf and it contains:

```
<Location /sb/>
  ProxyPass http://localhost:11111/
dest="pathmon=$SBJI:serverclass=SPRINGBOOT:mode=REQUEST_RESPONSE:pathsend_qualifier=**
CONTEXT-SENSITIVE**:single_dialog=true"
</Location>
```

- The /sb/ defines a prefix for the URL to make it unique in your environment. It specifies the URL as localhost with the port 11111 which just needs to be unique in the environment. It also includes the target PATHMON and SERVERCLASS

Spring Boot on NonStop

Where else can you deploy on the NonStop

- The curl command to call this NSHTTP copy of the application would be:

```
/usr/bin/curl  -X 'POST’ "http://10.10.69.130:8080/sb/doWork/" \
  -H 'accept: application/json'  -H 'Content-Type: application/json' \
  -d @doWork.json
```

- Where port 8080 is the port for the NSHTTP http daemon and the URL is added to match up with the sb.conf definition
- Output will be the same as direct access to the application:

```
{"empid":1,"firstname":"John","lastname":"NONSTOP_KERNEL","departid":1}
```

Spring Boot on NonStop

Where else can you deploy on the NonStop

- You can put the WAR file under NS SERVLETS FOR JAVASERVER PG (NSJSP) as a servlet. You can place the WAR file in the webapps directory of your NSJSP instance.
- Add an NSJSP instance
- Copy the Spring Boot WAR file to your servlets/webapps directory. In our test we renamed it sb.war
- Curl command to access this instance:

```
/usr/bin/curl  -X 'POST’  "http://10.10.69.130:8080/servlets/sb/doWork/" \
 -H 'accept: application/json' -H 'Content-Type: application/json' \
  -d @doWork.json
```

- Output:

```
{"empid":1,"firstname":"John","lastname":"NONSTOP_KERNEL","departid":1}
```

Spring Boot on NonStop

Where else can you deploy on the NonStop

- You can use NS API Gateway as the entry points for the microservice if it is using socket connections. There is another talk that discusses this in detail.

Thank you for attending this talk TBC23-TB60 Build a REpresentational State Transfer (REST) API microservice on NonStop in 15 minutes.

John Zimsky
John.Zimsky@HPE.com

HPE Slides and Materials Usage

This content is protected