

SDKMAN! का उपयोग कैसे करें

SDKMAN! (Software Development Kit Manager) एक शक्तिशाली उपकरण है जो आपको कमांड लाइन से एप्लिकेशन को तेजी से विकसित, टेस्ट और प्रोटोटाइप करने की अनुमति देता है। यह विशेष रूप से स्क्रिप्ट्स को चलाने के लिए एक पूर्ण प्रोजेक्ट संरचना स्थापित किए बिना, नए प्रोजेक्टों को जनरेट करने या फीचर्स के साथ प्रयोग करने के लिए उपयोगी है। नीचे SDKMAN! को प्रभावी रूप से इंस्टॉल और उपयोग करने का एक पूर्ण गाइड है।

इंस्टॉलेशन

SDKMAN! का उपयोग करने से पहले, आपको इसे इंस्टॉल करना होगा। आपके ऑपरेटिंग सिस्टम पर निर्भर करते हुए दो प्राथमिक विधियाँ हैं:

1. SDKMAN! का उपयोग (आधारित सिस्टम जैसे Linux या macOS के लिए अनुशंसित)

SDKMAN! एक उपकरण है जो सॉफ्टवेयर डेवलपमेंट किट्स को प्रबंधित करने के लिए है, जिससे Linux और macOS को इंस्टॉल करने का एक आसान तरीका बनता है।

कदम 1: SDKMAN! इंस्टॉल करें अपने टर्मिनल खोलें और निम्नलिखित कमांड चलाएं:

```
curl -s "https://get.sdkman.io" | bash
```

SDKMAN! को प्रारंभ करने के लिए स्क्रिप्ट को स्रोत करने के लिए प्रोम्प्ट्स का पालन करें:

```
source "$HOME/.sdkman/bin/sdkman-init.sh"
```

कदम 2: SDKMAN! इंस्टॉल करें निम्नलिखित कमांड चलाएं:

```
sdk install springboot
```

2. मैनुअल इंस्टॉलेशन (Windows या मैनुअल सेटअप के लिए)

अगर आप Windows पर हैं या मैनुअल इंस्टॉलेशन पसंद करते हैं: - ऑफिसियल SDKMAN! वेबसाइट से SDKMAN! बिनरी फ़ाइल डाउनलोड करें। - बिनरी फ़ाइल को अपने पसंद के डायरेक्टरी में एक्सट्रैक्ट करें। - एक्सट्रैक्टेड फ़ोल्डर से bin डायरेक्टरी को अपने सिस्टम के PATH पर्यावरण वरियेबल में जोड़ें।

इंस्टॉलेशन की पुष्टि

SDKMAN! बिनरी सही तरह से इंस्टॉल हो गया है कि यह सुनिश्चित करने के लिए, अपने टर्मिनल में निम्नलिखित कमांड चलाएं:

```
spring --version
```

आपको इंस्टॉल्ल्ड `spring` `cli` का संस्करण दिखना चाहिए (उदाहरण के लिए, Spring CLI v3.3.0). अगर यह काम करता है, तो आप इसे उपयोग करने के लिए तैयार हैं!

स्क्रिप्ट्स का उपयोग करने के प्रमुख तरीके

`spring` `cli` कई फीचर्स प्रदान करता है जो इसे तेजी से विकास और प्रोटोटाइपिंग के लिए आदर्श बनाते हैं। यहाँ इसके उपयोग करने के मुख्य तरीके हैं:

1. `spring` स्क्रिप्ट्स चलाना

`spring` `cli` का एक प्रमुख फीचर है कि यह `spring` स्क्रिप्ट्स को चलाने की अनुमति देता है बिना एक पूर्ण प्रोजेक्ट सेटअप की आवश्यकता। यह तेजी से प्रोटोटाइपिंग या `spring` `cli` के साथ प्रयोग करने के लिए आदर्श है।

- **उदाहरण: एक सरल वेब एप्लिकेशन बनाना** एक फ़ाइल बनाएं जिसका नाम `hello.groovy` है और निम्नलिखित सामग्री के साथ है:

```
@RestController
class HelloController {
    @RequestMapping("/")
    String home() {
        "Hello, World!"
    }
}
```

- **स्क्रिप्ट चलाएं** अपने टर्मिनल में `hello.groovy` को रखने वाले डायरेक्टरी में जाएं और निम्नलिखित कमांड चलाएं:

```
spring run hello.groovy
```

यह 8080 पोर्ट पर एक वेब सर्वर शुरू करता है। एक ब्राउज़र खोलें और `http://localhost:8080` पर जाएं ताकि "Hello, World!" दिखे।

- **डिपेंडेंसेज जोड़ना** आप स्क्रिप्ट में `@Grab` एनोटेशन का उपयोग करके डिपेंडेंसेज को सीधे शामिल कर सकते हैं। उदाहरण के लिए:

```
@Grab('org.springframework.boot:spring-boot-starter-data-jpa')
@RestController
class HelloController {
    @RequestMapping("/")
    String home() {
        "Hello, World!"
    }
}
```

यह आपके स्क्रिप्ट में `spring` `cli` को जोड़ता है बिना किसी बिल्ड फ़ाइल की आवश्यकता।

- **बहुत से स्क्रिप्ट्स चलाएं** वर्तमान डायरेक्टरी में सभी `spring` स्क्रिप्ट्स को चलाने के लिए, निम्नलिखित कमांड का उपयोग करें:

```
spring run *.groovy
```

2. नए प्रोजेक्ट बनाना

आपके पसंद के डिपेंडेंसेज के साथ एक नया प्रोजेक्ट संरचना जनरेट कर सकता है, जिससे आप एक पूर्ण एप्लिकेशन शुरू करने में समय बचा सकते हैं।

□ **उदाहरण: एक प्रोजेक्ट जनरेट करें** निम्नलिखित कमांड चलाएं ताकि वेब और डेटा-ज्पा डिपेंडेंसेज के साथ एक नया प्रोजेक्ट बनाएं:

```
spring init --dependencies=web,data-jpa my-project
```

यह एक डायरेक्टरी बनाता है जिसका नाम my-project है जिसमें `src/main/java` और `src/main/resources` के साथ कॉन्फिगर किए गए `src/main/resources` एप्लिकेशन है।

□ **कस्टमाइजेशन ऑप्शन** आप अतिरिक्त ऑप्शन जैसे निम्नलिखित को स्पेसिफाई कर सकते हैं:

- बिल्ड टूल: `--build=maven` या `--build=gradle`
- भाषा: `--language=java`, `--language=groovy`, या `--language=kotlin`
- पैकेजिंग: `--packaging=jar` या `--packaging=war`

उदाहरण के लिए:

```
spring init --dependencies=web --build=gradle --language=kotlin my-kotlin-project
```

3. एप्लिकेशन पैकेज करना

आपको अपने स्क्रिप्ट्स को डिप्लॉयमेंट के लिए एक्सीक्यूटेबल `jar` या `war` फ़ाइलों में पैकेज करने की अनुमति देता है।

□ **एक `jar` फ़ाइल बनाएं**

```
spring jar my-app.jar *.groovy
```

यह वर्तमान डायरेक्टरी में सभी `*.groovy` स्क्रिप्ट्स को my-app.jar में पैकेज करता है।

□ **एक `war` फ़ाइल बनाएं**

```
spring war my-app.war *.groovy
```

यह एक my-app.war फ़ाइल जनरेट करता है जो एक सर्वलेट कंटेनर में डिप्लॉय करने के लिए उपयुक्त है।

4. टेस्ट चलाना

अगर आपके पास `Test` स्क्रिप्ट हैं, तो आप उन्हें निम्नलिखित कमांड के साथ चला सकते हैं:

```
spring test *.groovy
```

यह वर्तमान डायरेक्टरी में सभी टेस्ट स्क्रिप्ट्स को चलाता है।

5. इंटरैक्टिव शेल का उपयोग

एक इंटरैक्टिव अनुभव के लिए, `spring shell` शेल लॉन्च करें:

```
spring shell
```

शेल के अंदर, आप कमांड जैसे `run`, `grab`, या `jar` इंटरैक्टिव रूप से चला सकते हैं, जो प्रयोग करने के लिए उत्तम है।

सारांश

`spring shell` एक उत्तम उपकरण है जो कमांड लाइन से `spring` के साथ तेजी से और दक्षता से काम करने वाले डेवलपर्स के लिए है। यहाँ इसे नटखट में उपयोग करने का तरीका है:

1. **इंस्टॉल करें** `spring shell` का उपयोग करके (`sdk install springboot`) या मैनुअल रूप से `spring` डाउनलोड करके और अपने `spring` को अपडेट करके।
2. **स्क्रिप्ट चलाएं** `spring run <script.groovy>` के साथ तेजी से प्रोटोटाइपिंग के लिए।
3. **नए प्रोजेक्ट बनाएं** `spring init --dependencies=<deps> <project-name>` के साथ।
4. **एप्लिकेशन को `jar` या `war` फ़ाइलों में पैकेज करें** `spring jar` या `spring war` का उपयोग करके।
5. **इंटरैक्टिव रूप से प्रयोग करें** `spring shell` के साथ।

चाहे आप एक विचार का प्रोटोटाइप बनाएं, `spring shell` सीख रहे हों या टास्क्स को ऑटोमेट करें, `spring shell` एक हल्का और फ्लेक्सिबल तरीका प्रदान करता है बिना किसी पूर्ण `spring` या प्रोजेक्ट सेटअप के ओवरहेड के शुरू करने के लिए।