Symfony - Practice Sheet

10 Q&A on DI, routing, Doctrine, events, security, and testing

Q1. What is Symfony Flex and how does it manage bundles?

A: Symfony Flex is a Composer plugin that maps package aliases (symfony/orm-pack) to recipes stored on a Flex server. On install it runs the recipe: creates config files, registers the bundle in bundles.php, and adds env vars to .env. This automates all boilerplate.

Q2. How does Symfony's DI container and autowiring work?

A: Symfony reads services.yaml and scans src/ for classes to register as services. Autowiring reads constructor type hints and injects matching services automatically. Interface bindings ($logger: Psr\Log\LoggerInterface) resolve abstractions to concrete

Q3. How does Symfony's routing system work?

A: #[Route('/users/{id}', name: 'user_show', methods: ['GET'])] on a controller method registers the route via PHP 8 attributes. Alternately use config/routes.yaml. Route::generate('user_show', ['id'=>1]) produces the URL. Prefix routes with #[Route('/api')] on the class.

Q4. What is Doctrine ORM and how does it differ from Active Record?

A: Doctrine uses the Data Mapper pattern: entities are plain PHP objects with no database knowledge. An EntityManager (separate object) handles persistence. Active Record (e.g., Eloquent) embeds DB logic in the model itself. Data Mapper enables richer domain models.

Q5. How does the Symfony EventDispatcher work?

A: Dispatch an event: $dispatcher->dispatch(new OrderCreated($order)). A subscriber implements EventSubscriberInterface and returns getSubscribedEvents(): [OrderCreated::class => 'onOrderCreated']. Tag the subscriber with kernel.event_subscriber for auto-registration.

Q6. How does Symfony Security's firewall system work?

A: A firewall in security.yaml defines which URLs it protects and how (form_login, jwt, api_key). The security layer authenticates the request and populates a token in the TokenStorage. Access control rules (access_control) check roles on top of the authenticated token.

Q7. How do Symfony Forms handle validation?

A: A FormType defines fields and maps them to a model class. $form->handleRequest($request) binds submitted data. $form->isSubmitted() && $form->isValid() checks Data Annotation (or YAML) constraints. $form->getErrors(true) returns validation messages.

Q8. How do you create a Symfony Console command?

A: Extend Command. Set the name in configure() via $this->setName('app:greet'). Define input arguments/options with addArgument/addOption. Implement execute($input, $output). Run with php bin/console app:greet --name=Alice. Tag with console.command for

Q9. How does Symfony Cache work with PSR-6?

A: Symfony's CacheInterface (a superset of PSR-6) provides get($key, callable). Use cache.adapter.redis in config to back the pool with Redis. @Cache(expires='1 hour') on a controller action caches the HTTP response via the HttpCache reverse proxy.

Q10. How do you write functional tests in Symfony?

A: Extend WebTestCase. $client = static::createClient(); $crawler = $client->request('GET', '/users/1'); $this->assertResponseIsSuccessful(). Use $client->clickLink() and $client->submitForm() to simulate user interaction. KernelTestCase is